FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Integração de processador ARC EM para aplicações HDCP 2.2

José Eduardo Ferreira Araújo

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: João Paulo de Castro Canas Ferreira

Supervisor Externo: Rui Sérgio Rainho de Almeida

28 de Julho de 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Integração de Processador ARC EM para Aplicações HDCP 2.2"

foi aprovada em provas realizadas em 17-07-2015

o júri

Presidente Professor Doutor José Alfredo Ribeiro da Silva Matos
Professor Catedrático do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Mário Pereira Véstias
Professor Coordenador do Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores do Instituto Superior de Engenharia de Lisboa

Professor Doutor João Paulo de Castro Canas Ferreira
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

José Eduardo Ferreira Araújo

Autor - José Eduardo Ferreira Araújo

Faculdade de Engenharia da Universidade do Porto

Resumo

A tecnologia HDMI é a interface multimédia com mais adoção para aplicações de entretenimento e multimédia. A mais recente especificação HDMI (High Definition Multimedia Interface 2.0) permite o transporte de resoluções de vídeo que podem chegar até 4Kx2K @ 60Hz (Ultra HD) e de áudio com frequências de amostragem até 1536KHz. De forma a proteger os conteúdos designados *premium* (eg. 4kx2K vídeo) contra cópia não autorizada, a interface HDMI permite a inclusão de tecnologia de encriptação HDCP2.2 (High-bandwidth Digital Content Protection).

Atualmente a solução HDCP 2.2 presente no portefólio Synopsys necessita que parte do processo de autenticação do protocolo HDCP 2.2 seja efetuado em *software* por um processador dedicado ou pelo processador de sistema. Esta solução torna mais difícil a definição do perímetro de segurança da solução HDCP 2.2 presente no DesignWare® HDMI RX/TX Interface IP da Synopsys. O objetivo deste trabalho passou por integrar um processador ARC® da família DesignWare® ARC® EM, na presente solução HDCP 2.2 do DesignWare® HDMI RX Interface IP da Synopsys por forma a dotar este produto da capacidade de operar, do ponto de vista de *hardware/software*, independente do processador de sistema, tornando assim a solução mais robusta, escalável e flexível.

Abstract

HDMI technology is the multimedia interface with more adoption for entertainment and multimedia applications. The latest HDMI specification (High Definition Multimedia Interface 2.0) allows the transport of video resolutions up to 4Kx2K @ 60Hz (Ultra HD) and audio at sampling frequencies up to 1536KHz. In order to protect the designated premium content (eg. 4kx2K video) against unauthorized copying, HDMI allows the inclusion of HDCP2.2 encryption technology (High-bandwidth Digital Content Protection).

The currently HDCP 2.2 solution in the Synopsys portfolio, requires that part of the HDCP 2.2 protocol authentication process is done in software by a dedicated processor or the system processor. This solution makes it difficult for the security perimeter definition of HDCP 2.2 solution present in DesignWare® HDMI RX / TX Interface IP, Synopsys. The project goal was to integrate a DesignWare® family ARC® processor ARC EM4 in the HDCP 2.2 solution of DesignWare® HDMI RX IP Interface. in order to give this product's ability to operate from the point of view for hardware/software, regardless of the system processor, thus making more robust, scalable and flexible solution.

Agradecimentos

Em primeiro lugar gostaria de agradecer à Faculdade de Engenharia da Universidade do Porto por todos os recursos e oportunidades que forneceu durante estes cinco anos como estudante, permitindo assim a conclusão desta dissertação assim como todo o curso de Engenharia Eletrotécnica e de Computadores.

Aos meus orientadores, Engenheiro Rui Sérgio Rainho de Almeida e Professor João Paulo de Castro Canas Ferreira por todo o apoio prestado durante esta dissertação e que me ajudarem a concluir a mesma.

À Synopsys, mais precisamente à equipa de HDMI pelo auxílio prestado durante a realização deste projeto.

Aos meus colegas da faculdade que me motivaram e ajudaram não só com este trabalho, mas também durante todos estes anos. Ao amigos da sala I222 e Biblioteca, ao pessoal de redes que também foram importantes assim como todo o pessoal que conheci durante toda a vida académica.

Às pessoas que tive um carinho muito especial, não só nos primeiros anos de faculdade, mas também aquelas que estavam presentes no inicio desta dissertação, bem como as que estão aquando o fim da mesma.

Aos amigos para a vida, designadamente o pessoal 12B, que estiveram e estarão sempre presentes seja para o que for. Já lá vão muitos anos de amizade e assim espero que continue.

Às pessoas mais importante da minha vida, a minha família. Sem eles, este percurso não era de todo possível. Aos meus pais, que para além de serem fantásticos, sempre me deram tudo o que tinham e que não tinham. Sempre me indicaram o melhor a nível académico mas principalmente a nível pessoal. A eles um eterno Obrigado! Ao meu primo Albano Araújo que me ajudou em tudo, não só com os seus conhecimentos, mas também com a sua experiência que me ajudou imenso a conseguir os meus objetivos. À minha prima, Ana Francisca Araújo, que também me apoiou bastante nesta vida, para além de estar presente em momentos especiais e por isso merece uma citação neste documento. Em geral um grande obrigado a toda a minha família.

Por fim, queria ainda agradecer ao meus familiares que já não estão entre nós, mas que continuarão sempre presentes.

José Eduardo Ferreira Araújo

"Avense ad eternum"

Conteúdo

1 Introdução **1**
- 1.1 Objetivos . . . 1
- 1.2 Contexto . . . 1
- 1.3 Motivação . . . 1
- 1.4 Contribuições . . . 2
- 1.5 Estrutura da Dissertação . . . 2

2 Enquadramento Técnico **3**
- 2.1 HDMI - *High-Definition Multimedia Interface* . . . 3
  - 2.1.1 Estrutura Principal . . . 4
- 2.2 HDCP - *High-bandwidth Digital Content Protection* . . . 4
  - 2.2.1 Processo de Autenticação . . . 4
- 2.3 Processador ARC EM4 . . . 5
- 2.4 Interfaces AMBA . . . 5
  - 2.4.1 AMBA APB3 . . . 6
  - 2.4.2 AMBA AHB-Lite . . . 7
- 2.5 Resumo . . . 8

3 Solução Existente **9**
- 3.1 Módulo Principal do Recetor HDMI . . . 9
- 3.2 Interface com protocolo SNPS HDCP2.2 . . . 10
- 3.3 Resumo . . . 11

4 Solução Proposta **13**
- 4.1 Localização da nova solução no atual sistema . . . 14
- 4.2 Estrutura Principal do Bloco *DWC_hdmi_rx_arcem* . . . 15
- 4.3 SNPS HDCP 2.2 uP . . . 15
- 4.4 Controlo ARC EM4 . . . 18
  - 4.4.1 Controlo do Processador ARC EM4 e geração de pedidos de interrupções 18
  - 4.4.2 Gestor das interrupções provenientes do processador ARC EM4 . . . 18
  - 4.4.3 Controlo do mecanismo de proteção de memória do processador ARC EM4 18
  - 4.4.4 Diagrama de Blocos . . . 19
- 4.5 APB Bridge . . . 21
- 4.6 APB 2 DTL Bridge . . . 22
- 4.7 APB 2 AHB-Lite Bridge . . . 22
- 4.8 AHB-Lite 2 APB Bridge . . . 23
- 4.9 Arquiteturas alternativas . . . 24
- 4.10 Espaço de Memória . . . 25

4.11 Descrição dos Registos . . . 26
4.11.1 Controlo ARC EM4 . . . 26
4.11.2 Memória DCCM (Memória de Dados) . . . 27
4.11.3 Memória ICCM (Memória de Instruções) . . . 28
4.11.4 Registos SNPS HDCP2.2 . . . 28
4.12 Fluxo de Operações . . . 29
4.13 Comandos do sistema . . . 33
4.13.1 Ligar/Desligar o bloco SNPS HDCP2.2 . . . 33
4.13.2 Obter o estado SNPS HDCP2.2 . . . 35
4.13.3 Reautenticação do bloco SNPS HDCP2.2 . . . 35
4.14 Software . . . 35
4.15 Resumo . . . 36

5 Simulação, Verificação e Resultados **39**
5.1 Introdução . . . 39
5.2 Plano de Verificação . . . 39
5.3 Simulação e Verificação . . . 39
5.4 Síntese em ASIC . . . 43
5.4.1 Resultados Temporais . . . 45
5.4.2 Área e Consumo de Potência . . . 46
5.4.3 Cadeias de Verificação . . . 46
5.4.4 *Clock Gating* . . . 47
5.5 Validação Formal . . . 47
5.6 Produção e Análise de Padrões de Teste . . . 48
5.7 Análise Temporal . . . 49
5.8 Síntese para FPGA . . . 49
5.9 Resumo . . . 51

6 Conclusão e Trabalhos Futuros **53**
6.1 Principais Conclusões do Trabalho . . . 53
6.2 Sugestões para Trabalhos Futuros . . . 53

A Plano de Verificação **55**

Referências **57**

Lista de Figuras

2.1 Exemplo de uma operação de escrita utilizando o protocolo AMBA APB3 [1] . . 6
2.2 Exemplo de uma operação de leitura utilizando o protocolo AMBA APB3 [1] . . 7
2.3 Exemplo de uma operação de leitura utilizando o protocolo AMBA AHB-Lite [2] 8
2.4 Exemplo de uma operação de escrita utilizando o protocolo AMBA AHB-Lite [2] 8

3.1 Exemplo de interfaces que constituem um sistema multimédia [3] 10
3.2 Esquemático da solução existente 11

4.1 Localização do módulo *DWC_hdmi_rx_arcem* no sistema 14
4.2 Diagrama de blocos do módulo *DWC_hdmi_rx_arcem* 16
4.3 Esquema do modo de operação das interrupções 19
4.4 Diagrama de blocos do módulo *DWC_hdmi_rx_arcem_cfgshell* 20
4.5 Diagrama de blocos do módulo *DWC_hdmi_rx_arcem_cfgshell* 20
4.6 Estrutura do módulo *DWC_hdmi_rx_arcem_apb_bridge* 21
4.7 Máquina de estados do bloco *DWC_hdmi_rx_arcem_apb2dtl* 22
4.8 Máquina de estados que controla o módulo AMBA APB3 2ahblite 23
4.9 Estrutura do módulo *DWC_hdmi_rx_arcem_apb2ahblite* 24
4.10 Mapaeamento da memória utilizado no bloco *DWC_hdmi_rx_arcem* 25
4.11 Fluxo de operações para o envio de um comando 31
4.12 Fluxo de operações para a receção de um comando 32
4.13 Sequência de operações para ligar o bloco SNPS HDCP2.2 34
4.14 Sequência de operações para desligar o bloco SNPS HDCP2.2 34
4.15 Sequência de operações para obter o estado do bloco SNPS HDCP2.2 35
4.16 Sequência de operações para reautenticar o bloco SNPS HDCP2.2 36

5.1 Formas de Onda de operações utilizando o acesso AMBA APB3 40
5.2 Formas de onda de acessos para envios de Interrupções 41
5.3 Formas de onda na receção de Interrupções 42
5.4 *Code Coverage* do módulo *DWC_hdmi_rx_arcem* 43
5.5 Fluxo da ferramenta *DFT Compiler* [4] 44
5.6 Relatório do número de elementos que utilizam *Clock Gating* 47
5.7 Fluxo da ferramenta *TetraMax* [5] 48

Lista de Tabelas

4.1 Lista dos bloco que compõem o módulo *DWC_hdmi_rx_arcem* 17
4.2 Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* 27
4.3 Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* referente às interrupções 27
4.4 Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* referentes ao controlo da proteção das memórias 28
4.5 Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* 29
4.6 Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* referente às interrupções 29
4.7 Bits que indicam o tipo de comando 33
4.8 Bits que indicam o tipo de comando 34

5.1 *Slacks* obtidos na Síntese em ASIC do módulo *DWC_hdmi_rx_arcem* 45
5.2 Frequências de relógio obtidas na Síntese em ASIC do módulo *DWC_hdmi_rx_arcem* 46
5.3 Área e consumos de potência obtidos na síntese em ASIC 46
5.4 Falhas do tipo *Stuck* 49
5.5 Falhas do tipo *Transient* 49
5.6 Restrições temporais das entradas e saídas utilizadas para a síntese em FPGA . . 50
5.7 Frequências obtidas na síntese em FPGA 50
5.8 Área ocupada pelo módulo *DWC_hdmi_rx_arcem* na síntese em FPGA 50

A.1 Plano de Verificação usado no módulo *DWC_hdmi_rx_arcem* 56

Abreviaturas e Símbolos

AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
ASIC	Application Specific Integrated Circuits
AXI	Advanced eXtensible Interface
CBUS	Control Bus
DCCM	Data Closely Coupled Memories
DDC	Display Data Channel
DFT	Design For Test
DVE	Discovery Visual Environment
DVI	Digital Visual Interface
FIFO	First in First Out
FPGA	Field-Programmable Gate Array
HDCP	High-bandwidth Digital Content Protection
HDMI	High-Definition Multimedia Interface
HDMI RX	Recetor HDMI
I^2C	Inter-Integrated Circuit
ICCM	Instruction Closely Coupled Memories
ISA	Instruction Set Architecture
KSV	Key Selection Vector
MHL	Mobile High-Definition Link
MIPI	Mobile Industry Processor Interface
SNPS	Synopsys
SoC	System on Chip
TDMS	Transition Minimized Differential Signaling
uP	Microprocessador
VCS	Verilog Compiled code Simulator

Capítulo 1

Introdução

Neste capítulo será realizada toda a contextualização deste relatório, a motivação que levou à realização de todo este projeto, bem como a estrutura do presente documento.

1.1 Objetivos

O objetivo deste trabalho é proceder à integração de um processador ARC®, da família DesignWare® ARC EM4, na solução HDCP 2.2 (High-bandwidth Digital Content Protection System 2.2) presente nos produtos DesignWare® HDMI RX/TX Interface IP da Synopsys.

A implementação deverá utilizar linguagem de descrição de *hardware Verilog* e ser sintetizável nas tecnologias 40nm e 28nm. O propósito final deste trabalho será dotar a solução HDCP 2.2 da Synopsys da capacidade de operar, do ponto de vista de *Hardware/Software*, independentemente do processador de sistema, simplificando, desta forma, a definição do perímetro de segurança à volta da estratégia HDCP 2.2 dos produtos DesignWare® HDMI RX/TX Interface IP da Synopsys.

1.2 Contexto

A Synopsys, referenciada anteriormente, é uma empresa multinacional que trabalha essencialmente em projetos de circuitos integrados e soluções para os mesmos. A Synopsys Porto, localizada na cidade da Maia, é responsável por conceber interfaces ao nível do HDMI, MHL e até MIPI. Essa dissertação, proposta pela Synopsys Porto, enquadra-se na solução HDMI, mais especificamente na segurança dos dados do recetor HDMI (HDMI RX), tendo em vista uma melhoria no desempenho geral do sistema.

1.3 Motivação

Nos dias que correm, os serviços multimédia estão bastante presentes no quotidiano das pessoas sendo, por isso, importante mantê-los com um nível de qualidade elevado. Para além deste

parâmetro fulcral, é também necessário manter um certo nível de segurança contra vários tipos de ameaças como pirataria e cópias ilegais de conteúdos.

Do ponto de vista do utilizador final, é importante que estes sistemas sejam rápidos, robustos e de grande fiabilidade. Para isso, é imperativa a existência de uma interface com os sistemas, que seja fácil de manipular e de configurar, mas que, ao mesmo tempo, mantenha ao seu dispor todas as suas funcionalidades. Com este trabalho pretende-se então adicionar na solução atual, uma estrutura que forneça ao utilizador final uma interface com o protocolo de segurança de dados, que seja simples mas ao mesmo tempo robusta e que mantenha um grande nível de seguridade.

1.4 Contribuições

Todo este projeto está incluído na implementação de toda a interface HDMI, realizada pela Synopsys. Assim sendo, esta dissertação pretende integrar todos os resultados obtidos no atual portfólio da Synopsys, contribuindo assim para uma melhoria da solução atual, mais precisamente no acesso ao protocolo de segurança SNPS HDCP2.2.

1.5 Estrutura da Dissertação

Esta dissertação encontra-se dividida em 6 capítulos. No primeiro capítulo é realizada a introdução à temática discutida nesta dissertação, abordando-se também a motivação que levou à sua concretização.

No segundo capítulo é realizado o enquadramento técnico da dissertação, explicando e abordando todos os aspetos e conhecimentos que irão ser usados em toda a dissertação.

No terceiro capítulo será explicada a atual solução existente no portefólio da Synopsys e quais as suas desvantagens.

No quarto capítulo será apresentada a solução que irá ser acrescentada no atual sistema, para que o mesmo cumpra os objetivos desta dissertação.

No quinto capítulo serão apresentados os resultados obtidos após a implementação, simulação e verificação do módulo apresentado no capítulo quatro.

A dissertação encerra no sexto capítulo, onde são realizadas as conclusões desta dissertação e expostas algumas propostas para desenvolvimentos futuros deste projeto.

Capítulo 2

Enquadramento Técnico

Neste capítulo é feita uma breve introdução teórica sobre todos os aspetos tratados neste relatório.

2.1 HDMI - *High-Definition Multimedia Interface*

O HDMI é uma interface de comunicação totalmente digital tipicamente usada para transmitir vídeo de alta resolução em conjunto com áudio. Hoje em dia, a interface HDMI é amplamente usada e está presente na maior parte dos dispositivos que trabalham com áudio/vídeo digital tais como televisões, computadores, consolas de jogos, set-top boxes, entre muitos outros.

Numa comunicação HDMI é necessário conter sempre um transmissor e um recetor, sendo que por vezes é possível intercalar com alguns repetidores. Porém, este último elemento, pode ser visto como um dispositivo que faz de recetor e transmissor em simultâneo. Estes repetidores têm como principal função retransmitir o sinal recebido.

Desde do aparecimento da interface HDMI que esta tem sofrido várias revisões, encontrando-se neste momento na revisão 2.0. Para além das características existentes nas revisões anteriores, esta permite uma velocidade de transmissão de 6 Gbit/s por canal, o que possibilita o envio de vídeo com uma resolução de 2160p (4K) a 60 *frames* por segundo. Relativamente à transmissão de áudio, esta revisão permite o envio de até 32 canais de áudio possibilitando assim uma melhor experiência a nível sonoro. O HDMI também oferece a possibilidade de transmitir comandos que controlam os dispositivos ligados por HDMI, como por exemplo, alterar o som do televisor através do comando remoto de uma *set-top box* [6].

Comparando o HDMI com as outras interfaces existentes no mercado, como é o caso do DVI (*Digital Visual Interface*) e do Vídeo-Componente, verifica-se que a principal vantagem do HDMI é a transmissão de áudio no mesmo cabo, funcionalidade que não está presente no seus concorrentes. O HDMI é puramente digital e neste momento já permite grandes resoluções de vídeo, ao invés das outras interfaces [6].

2.1.1 Estrutura Principal

Um cabo HDMI é composto por quatro pares diferenciais que são usados para transmitir todos os dados (vídeo, áudio e dados auxiliares), assim como o sinal de relógio. Estes canais são designados por canais TDMS e têm a particularidade de codificar os 8 bits a serem transmitidos numa palavra de 10 bits com características que permitem uma transmissão mais eficiente. Para além destas ligações também existem outras que servem para transmitir configurações e estados da comunicação [7].

2.2 HDCP - *High-bandwidth Digital Content Protection*

Dentro da interface HDMI existe um protocolo de encriptação que incorpora alguma segurança no sistema. Este protocolo é designado de HDCP (High-bandwidth Digital Content Protection) e é desenhado para proteger as transmissões de audiovisuais de serem gravadas e copiadas por dispositivos externos. Cada dispositivo que utilize este protocolo contém um conjunto de chaves secretas e um identificador (proveniente da *Digital Content Protection LLC*). Este identificador é designado por KSV (*Key Selection Vector*) e é composto por 40 bits [8].

Existem três elementos essenciais no sistema de proteção. O primeiro é o processo de autenticação realizado pelo transmissor e que tem o intuito de verificar se o recetor está em condições de receber os dados de transmissão. De seguida, é iniciada a transferência dos dados encriptados entre o transmissor e o recetor, baseando-se nas chaves partilhadas entre os dois dispositivos. Já o último elemento corresponde à constante monitorização da legitimidade dos dispositivos envolvidos na transmissão, para que em algum momento o envio dos dados seja interrompido caso o transmissor identifique que um dos dispositivos esteja comprometido (ou seja, deixou de obedecer as especificações do protocolo) [8].

2.2.1 Processo de Autenticação

O processo de autenticação no protocolo HDCP é dividido em três fases principais. A primeira fase ocorre logo após a conexão entre os dois dispositivos. O transmissor começa por enviar ao recetor um código de 64 bits gerado aleatoriamente, em conjunto com o seu identificador KSV. De seguida, o recetor responde com o seu KSV juntamente com um bit que assinala se é repetidor ou se é apenas um recetor. A partir deste momento os dois dispositivos efetuam operações matemáticas entre os dois KSVs e o código gerado pelo transmissor, sendo que o resultado final tem de ser o mesmo nos dois dispositivos. Se assim acontecer, então a primeira fase de autenticação está concluída; caso contrário, a transmissão é rejeitada [8].

A segunda fase de autenticação do protocolo HDCP é apenas realizada se o dispositivo recetor for um repetidor caso o bit de repetidor, que foi referenciado na primeira fase, for igual a 1. Em caso afirmativo, o repetidor recolhe as KSVs dos dispositivos que estão a jusante, efetua mais operações matemáticas com todos esses códigos e utilizando um conjunto de condições verifica

se todos os dispositivos são legítimos. Se sim, então o processo de autenticação continua; caso contrário, a transmissão é cancelada [8].

Por fim, a terceira e última fase de autenticação é muito idêntica à primeira. Esta autenticação é sempre realizada antes do envio de cada *frame* de vídeo e qualquer erro no cálculo das chaves ou discrepância do protocolo é entendido pelo transmissor como uma violação dos dados, sendo a transmissão interrompida [8].

2.3 Processador ARC EM4

O ARC EM4 da DesignWare [9] é um processador de dimensões reduzidas, com alta performance e com um baixo consumo de potência. É amplamente usado em sistemas embebidos assim como sensores, memórias ou até em sistemas que usam baterias como fonte de alimentação.

Este processador é baseado na nova ISA ARCv2 de 32-bits, o que leva a melhorias significativas no seu desempenho, assim como na redução da densidade de código em cerca de 20%. Esta nova arquitetura, juntamente com o *pipeline*, levam a que o processador corresponda às necessidades da maior parte das aplicações SoC, sendo elas de baixo consumo de potência ou então de elevada performance.

O ARC EM4 suporta uma vasta gama de opções configuráveis, sendo possível inserir ou remover especificações, otimizando assim o seu desempenho para cada aplicação específica. Este dispositivo suporta memórias encapsuladas (sejam de dados ou de instruções) que podem ir de 512B até 1MB, além do facto de permitirem acessos vindos do exterior, usando para tal, uma interface AMBA AHB-Lite. Apesar de as memórias apenas suportarem esta interface, o processador em si permite outras interfaces de barramentos, tais como BVCI, o que leva a uma melhor integração com outros dispositivos. Outras características importantes presentes neste processador passam pela presença de multiplicadores de 16x16 e 32x32 bits, assim como a implementação de interrupções e de exceções, com o suporte de prioridades.

Este processador é configurável através de uma ferramenta designada de *ARChitect2*. Este *Software* contém uma interface gráfica que permite de forma intuitiva acrescentar funcionalidades ao processador. Após a correta configuração do processador, esta ferramenta produz código Verilog que pode ser usado e integrado em qualquer projeto. Esta ferramenta produz também testes específicos em *hardware* que permitem verificar que o módulo gerado está correto e que opera como era esperado.

2.4 Interfaces AMBA

A arquitetura AMBA (*Advanced Microcontroller Bus Architecture*) é uma especificação usada em conexões de blocos funcionais. Esta interface é amplamente usada em implementações de controladores e periféricos.

Esta tecnologia tem como objetivo interligar vários componentes de uma forma fácil e eficaz sem necessidade de implementação de protocolos de comunicação. É independente da tecnologia

usada no sistema e é altamente recomendada para implementações modulares, sendo esta arquitetura usada para na comunicação entre os vários módulos.

Esta interface divide-se em várias modelos específicos para cada tipo de implementação. Na arquitetura do bloco correspondente ao HDMI é bastante usada a interface AMBA APB3. Este tipo de interface é usado em comunicações que utilizam pouca largura de banda como por exemplo no acesso a registos. Esta interface está explica com mais detalhe na secção seguinte.

2.4.1 AMBA APB3

O protocolo AMBA APB3 [1] fornece uma interface de baixo custo com um consumo mínimo de potência e com uma complexidade reduzida. Este protocolo é constituído por vários sinais sendo que estes são divididos em dois grupos, sinais de controlo e sinais de dados.

No que diz respeito aos sinais de dados, existem os dados de escrita e os dados de leitura (*PWDATA* e *PRDATA* respetivamente).

Relativamente aos sinais de controlo, existem os sinais de *SEL* e *ENABLE* que indicam o estado da operação, sendo que o sinal de *WRITE* indica se é uma operação de escrita ou de leitura. Para finalizar a operação existem os sinais de *READY* e de *SLVERR* que indicam respetivamente o fim da operação e um possível erro que possa ter ocorrido no acesso ao módulo de destino. Nas Figuras 2.1 e 2.2 estão representados estes mesmos sinais apesar de os nomes estarem com o prefixo *P*, o que indica que os mesmos pertencerem ao protocolo AMBA APB3.

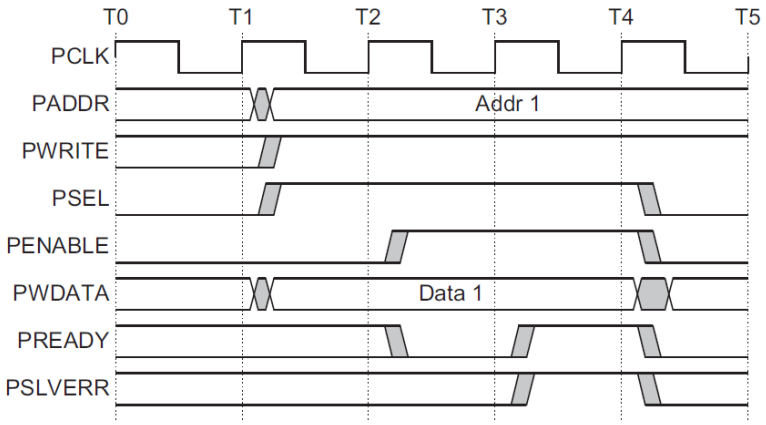

Figura 2.1: Exemplo de uma operação de escrita utilizando o protocolo AMBA APB3 [1]

Na Figura 2.1 está representado uma transferência utilizando o protocolo AMBA APB3. Nesta operação está referenciada uma escrita no endereço *Addr 1* com os dados *Data 1*, mas que a mesma retornou um erro indicado pela ativação do sinal *PSLVERR*. No caso de não ter ocorrido nenhum erro, as formas de onda eram idênticas às da Figura 2.1, com exceção do sinal *PSLVERR* que se mantém no nível lógico 0.

Já no caso de ser uma leitura, as formas de ondas têm o mesmo formato da operação de escrita estando a diferença no sinal *PWRITE*. Esta operação está apresentada na Figura 2.2.

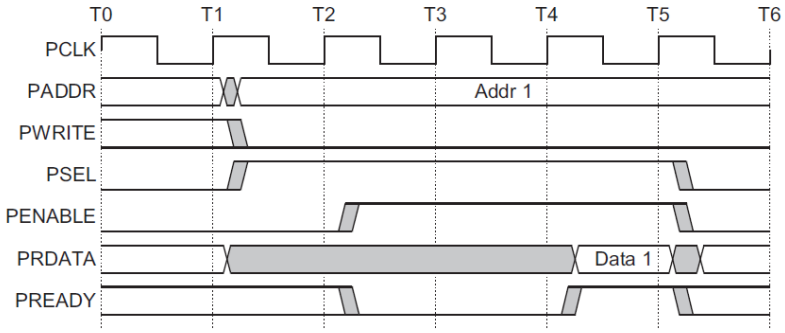

Figura 2.2: Exemplo de uma operação de leitura utilizando o protocolo AMBA APB3 [1]

Neste caso os dados que estão associados ao endereço *Addr 1* surgem aquando a ativação do sinal que indica o final da operação, ou seja, o sinal *PREADY*.

No caso de existir um erro na operação, o formato é idêntico, apenas com a diferença que o sinal *PSLVERR* é ativado aquando o *PREADY*.

2.4.2 AMBA AHB-Lite

Outra tipo de interface AMBA muito usado em projetos de circuitos digitais é o protocolo AMBA AHB-Lite [2]. Este interface é a que o processador ARC EM4 oferece para o acesso às memórias assim como a entidades externas.

Este tipo de interface é uma versão mais robusta do protocolo AMBA APB3. Utiliza apenas dois ciclos de relógio para realizar uma operação completa (assumindo que não existem estados de espera) e permite transmissões consecutivas (modo *burst*) sem que o sinal de seleção seja desativado. A versão AMBA AHB-Lite, comparativamente à sua versão original (AMBA AHB), apenas permite um *master*.

Para além dos sinais que existem no protocolo AMBA APB3, esta interface é contemplada também com outros sinais de controlo. Estes sinais permitem configurar operações especais suportadas pelo protocolo AMBA AHB-Lite, como operações em sequência, bloqueios de operações, operações com uma menor quantidade de dados entre outros exemplos.

Nas Figuras 2.3 e 2.4 estão representadas as formas de onda de transferências utilizando o protocolo AMBA AHB-Lite.

Relativamente à operação de leitura é possível observar que os dados associados ao endereço *A* surgem no ciclo de relógio seguinte, o que torna este protocolo bastante rápido e eficiente. O mesmo acontece no caso de uma operação de escrita, representada na Figura 2.4, os dados a enviar para o endereço *A* são colocados na linha no ciclo de relógio seguinte, enquanto o endereço já pode ser mudado para uma nova operação.

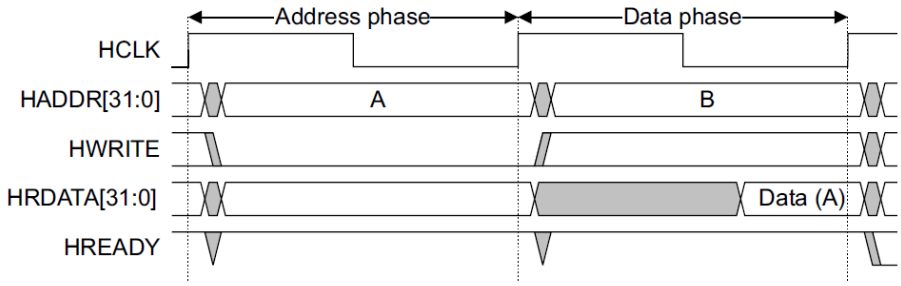

Figura 2.3: Exemplo de uma operação de leitura utilizando o protocolo AMBA AHB-Lite [2]

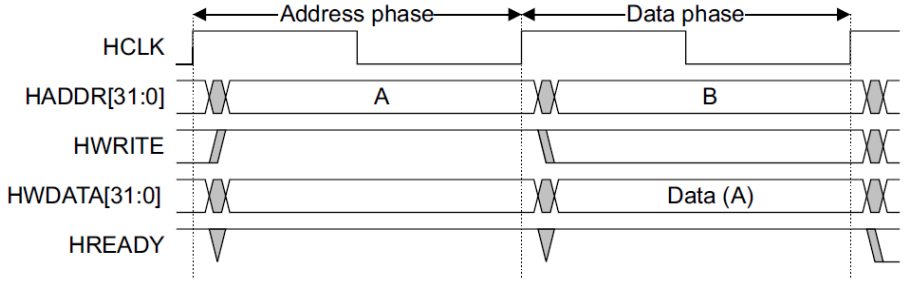

Figura 2.4: Exemplo de uma operação de escrita utilizando o protocolo AMBA AHB-Lite [2]

2.5 Resumo

Neste capitulo foi introduzido os termos teóricos usados nesta dissertação, assim como o estado da arte dos principais aspetos utilizados. Estes termos, aqui referenciados, vão ser usados nos próximos capítulos para descrever os problema da dissertação.

Capítulo 3

Solução Existente

Neste capítulo é apresentado todo o problema existente à volta da dissertação, quais as suas desvantagens e o porque da necessidade da alteração da atual solução.

3.1 Módulo Principal do Recetor HDMI

Na presente solução da Synopsys, todo o controlo da interface HDMI é feito pelo processador do sistema, o que implica um esforço adicional por parte do mesmo. O processador necessita de ler dados do controlador HDMI, processa-los e escrever os resultados nos registos do controlador, o que conduz a que o processador despenda de recursos temporais importantes para outras tarefas, para além de diminuir o nível de segurança do sistema.

O módulo principal, no qual se insere todo este trabalho, é designado por *DWC_hdmi_rx* [3]. Este módulo é ligado ao SoC principal, onde estão conectadas outras interfaces, como está representado na Figura 3.1. Aqui são apresentadas todas a interfaces que o processador do sistema tem de sustentar para além do controlo do HDMI.

Este módulo tem como objetivo separar todos os dados recebidos, sejam eles dados de vídeo, dados de áudio ou dados de configuração. Este controlador consegue configura-se automaticamente, baseando-se nos dados de configuração recebidos, sem necessidade de haver uma intervenção de *software*. É composto por vários componentes mais pequenos, necessários para que todo o protocolo funcione corretamente. Entre eles, é possível destacar o descodificador TDMS que está encarregue de processar os dados recebidos e transforma-los em informação percetível ao recetor. Outro componente importante são as memórias FIFO responsáveis por guardar todos os dados recebidos, assim como os KSVs, do protocolo HDCP, explicados anteriormente neste documento. Para suportar a comunicação entre os módulos HDCP do transmissor e do recetor é um usado um canal DDC, o qual é baseado no protocolo I^2C. Como este protocolo baseia-se em comunicação endereçada, torna-se mais fácil a escrita de informação por parte do transmissor, nas memórias do recetor.

Dentro do controlador HDMI, está presente outro bloco importante, designado por CBUS, responsável pelo controlo do módulo principal e do protocolo HDCP. Este módulo está presente

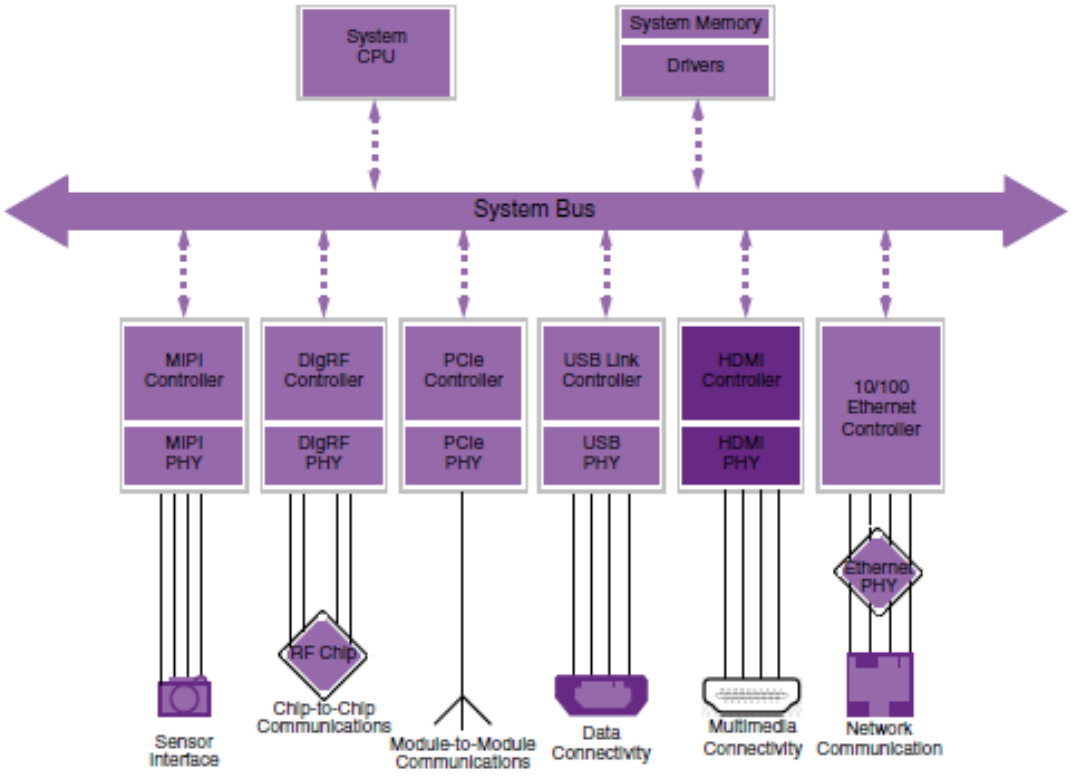

Figura 3.1: Exemplo de interfaces que constituem um sistema multimédia [3]

quando existe uma conexão com a interface MHL, e em caso contrário, estas operações são realizadas no módulo principal.

O processador do sistema comunica com os registos internos através de um barramento do tipo AMBA APB3. Neste barramento é passado qualquer dado de controlo do sistema assim como as interrupções relativas a determinados eventos.

Como referido anteriormente, a interface HDMI contém um protocolo de segurança de dados designado de SNPS HDCP2.2, que protege a informação transmitida de ser copiada. Este bloco contém uma interface própria com o exterior, que permite um acesso muito mais rápido e controlado por parte do utilizador final.

3.2 Interface com protocolo SNPS HDCP2.2

A atual interface com o protocolo SNPS HDCP2.2 que está presente nos controladores HDMI, baseia-se numa conexão direta entre o processador do sistema e os registos de configuração. Estes registos, que permitem a configuração de todo o protocolo SNPS HDCP2.2, estão inseridos dentro do bloco *h22_cfgshell* (representado a azul na parte inferior da Figura 3.2). Como é possível verificar esta ligação é feita através de uma interface AMBA APB3 juntamente com uma linha de interrupção. Todo esta solução está apresentada na Figura 3.2.

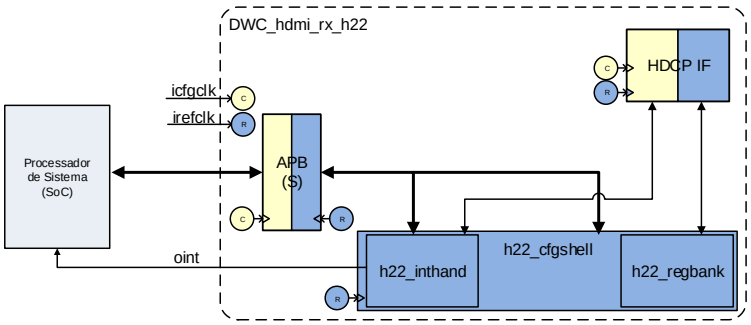

Figura 3.2: Esquemático da solução existente

Quando é necessário efetuar qualquer configuração no protocolo SNPS HDCP2.2, o processador do sistema tem efetuar escritas em registos específicos. Quando é gerado algum evento no bloco do SNPS HDCP2.2 (como exemplo, uma memória FIFO fica totalmente preenchida), o mesmo despoleta uma interrupção que informará o processador de sistema. Por sua vez, o processador precisa de realizar uma leitura nos registos correspondentes as interrupções para verificar qual o motivo da mesma, para além de tomar as medidas necessárias com base no acontecimento ocorrido. Este sistema é facilmente acedido por fontes não desejadas, uma vez que basta conhecer o endereço dos registos de configuração para colocar o protocolo SNPS HDCP2.2 em qualquer estado, ficando a segurança do protocolo comprometida.

O intuito deste projeto é adicionar uma barreira entre a interface AMBA APB3 que existe atualmente e o bloco que contém os registos relativos ao SNPS HDCP2.2. Assim o processador de sistema não ficará com acesso direto aos registos do protocolo de encriptação, aumentado o perímetros de segurança e retirando carga ao processador principal.

3.3 Resumo

Neste capitulo foi apresentado a solução existente no portfólio da Synopsys, bem como o modo que o processador do sistema atua na configuração do bloco SNPS HDCP2.2.

Capítulo 4

Solução Proposta

A solução existente e apresentada no capítulo anterior usava uma ligação direta entre o processador de sistema e o bloco SNPS HDCP2.2. O objetivo deste projeto é a inserção de um bloco funcional que substitui a conexão existente. Este bloco funcional será constituído por um processador ARC EM4 e por uma lógica envolvente ao mesmo, que permitirá a troca de informações e comandos entre o processador de sistema e o protocolo SNPS HDCP2.2.

Com este novo módulo, o nível de segurança da encriptação SNPS HDCP2.2 irá aumentar, uma vez que será necessário conhecer todos as etapas necessárias para aceder ao protocolo, assim como todos os endereços definidos para se conseguir aceder ao bloco do SNPS HDCP2.2, ao contrário do que acontecia na solução anterior, em que o processador do sistema tinha acesso aos registos de informação e configuração do protocolo de encriptação.

A nova solução a ser construída necessita de cumprir algumas exigências iniciais que servirão de base para a escolha da melhor implementação. Estas condições passam por proteger todos os dados relativos ao SNPS HDCP2.2, isto é, não dar acesso direto ao processador de sistema. O utilizador final apenas poderá dizer ao novo sistema que quer realizar determinada operação no controlador SNPS HDCP2.2 e por consequente, receber algum *feedback* sobre o estado dessa mesma operação.

Uma vez que o objetivo do projeto é retirar esforço do processador do sistema e entrega-lo ao novo processador ARC EM4, será este último que irá ter a tarefa de aceder às informações e registos do protocolo SNPS HDCP2.2. Deste modo, implicará que de alguma forma o utilizador comunique com o processador ARC EM4, de maneira a indicar que operação deseja realizar no bloco SNPS HDCP2.2.

A solução encontrada é uma arquitetura baseada em *MailBoxes*, isto é, existe um espaço de memória que é acessível pelas duas entidades e que vai ser usado para transmitir informação entre as mesmas. Com esta implementação, o utilizador poderá enviar ao processador ARC EM4 o tipo de operação que deseja realizar, assim como vários dados associados. Uma vez que a maioria das operações a realizar no controlador SNPS HDCP2.2 implicam uma resposta por parte do mesmo, é necessário dividir o espaço de memória acima referido em duas partes distintas, para que seja possível a transmissão de pedidos e respostas entre as duas entidades de uma forma mais

organizada e sem interferências.

Quando se utiliza memória partilhadas é importante também ter atenção aos acessos mútuos ao mesmo local, isto é, duas entidades diferentes tentarem aceder a mesma memória ao mesmo tempo. Assim sendo, foi necessário implementar um mecanismo que apenas permita um acesso de cada vez às respetivas *MailBoxes*. Foi escolhido um mecanismo de interrupções no sistema, uma vez que nos tempos atuais, a maioria dos processadores oferecem uma interface com interrupções muito intuitiva e prática. As interrupções representam uma série de pedidos e repostas por parte das duas entidades, para que o acesso às memórias fique muito mais controlado, tornando assim o sistema mais eficaz.

Resumindo todos estes requisitos, a base de operação do sistema irá passar pelo envio de comandos e respetivas respostas através de memórias partilhadas, sendo estas controladas por interrupções entre os dois processadores.

4.1 Localização da nova solução no atual sistema

Toda a lógica necessária para a correta operação do sistema, assim como o novo processador estão inseridos num bloco designado por *DWC_hdmi_rx_arcem*, o qual vai estar em paralelo com a atual solução do HDMI RX (Recetor HDMI) presente na Synopsys. Por sua vez, estes dois módulos estão inseridos no bloco de topo designado por *DWC_hdmi_rx_ctrl*. Na Figura 4.1 é apresentada a localização no sistema do novo bloco.

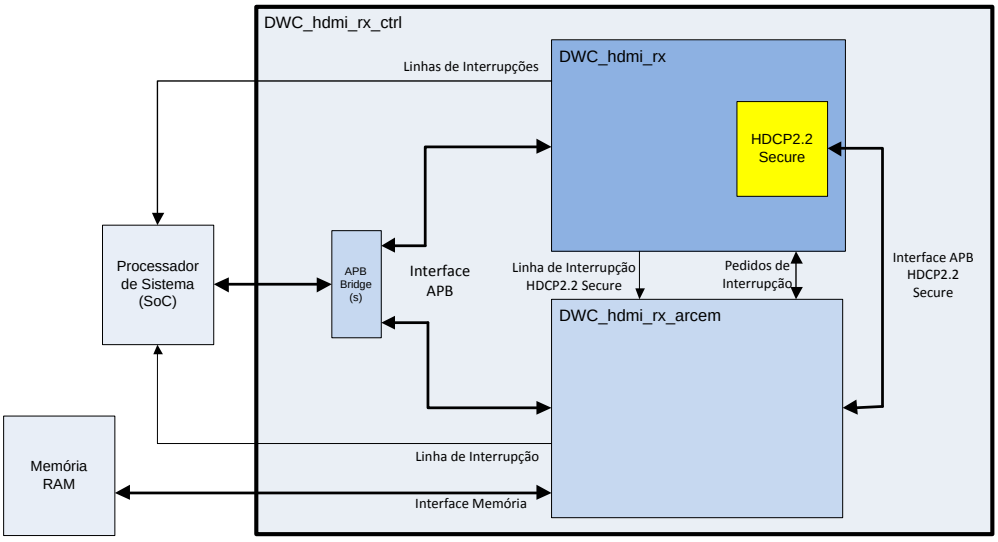

Figura 4.1: Localização do módulo *DWC_hdmi_rx_arcem* no sistema

Entre o novo bloco (*DWC_hdmi_rx_arcem*) e o resto do sistema irão existir ligações que permitirão a comunicação com o processador de sistema assim como o protocolo SNPS HDCP2.2.

Para além destas ligações, existirão também linhas de interrupções entre os diversos blocos que compõem o sistema. Todas estas ligações, que estão representadas na Figura 4.1, vão permitir uma comunicação bem estruturada com o processador e a atual solução do HDMI RX.

Uma vez que a interface do processador do sistema com o HDMI RX é do tipo AMBA APB3, optou-se por colocar uma interface do mesmo tipo no acesso ao bloco que incluirá o processador ARC EM4, ficando assim uma ligação mais uniformizada. O mesmo acontece com o acesso ao bloco do SNPS HDCP2.2, ou seja, a interface com o mesmo é do tipo AMBA APB3, o que obriga o módulo *DWC_hdmi_rx_arcem* a ter mais um barramento AMBA APB3.

Como explicado anteriormente, existirão também linhas de interrupções entre os dois processadores. Assim sendo, existirão dois fluxos de interrupções entre as duas entidades, um proveniente do bloco *DWC_hdmi_rx_arcem* com destino ao processador do sistema e outro com destino ao novo processador presente no bloco *DWC_hdmi_rx_arcem*. Cada fluxo de interrupção será constituído por registos que despoletam pedidos de interrupção sendo que estes pedidos serão depois unidos através de operações lógicas, onde se vai formar a linha de interrupção que se liga à entidade de destino.

4.2 Estrutura Principal do Bloco *DWC_hdmi_rx_arcem*

Maioritariamente, este projeto passa pela construção e validação de um bloco que cumpra as especificações anteriormente apresentadas. Na Figura 4.2 está apresentado o diagrama de blocos do módulo *DWC_hdmi_rx_arcem* com as respetivas ligações entre os mesmos.

Neste bloco destacam-se dois módulos principais que realizam a maior parte das funcionalidades do sistema. O primeiro corresponde ao novo processador ARC EM4, já o segundo corresponde a um bloco de controlo do novo processador, o qual tem o nome de *DWC_hdmi_rx_arcem_cfgshell*.

Para a comunicação com o processador de sistema é necessário uma interface AMBA APB3, como explicado anteriormente, a qual é sustentada pelo bloco APB Bridge. Este bloco reencaminha as operações de escrita e leitura para os diversos blocos de registos e memórias dependendo do endereço de entrada. Estão também presentes blocos que possibilitam a conversão de uma interface para outra, assim como o domínio de relógio. Por fim existem ainda sincronizadores de pulsos cujo seu objetivo é permitir que um pulso seja transferido para um domínio de relógio diferente do domínio original.

Na Tabela 4.1 estão apresentados todos os blocos utilizados na construção deste sistema assim como a sua respetiva funcionalidade.

Cada bloco está explicado com mais detalhe nas secções seguintes.

4.3 SNPS HDCP 2.2 uP

É neste módulo que estará inserido o processador ARC EM4 [9] que irá remover todo o esforço relativo ao protocolo SNPS HDCP2.2. Este processador é configurável através da ferramenta

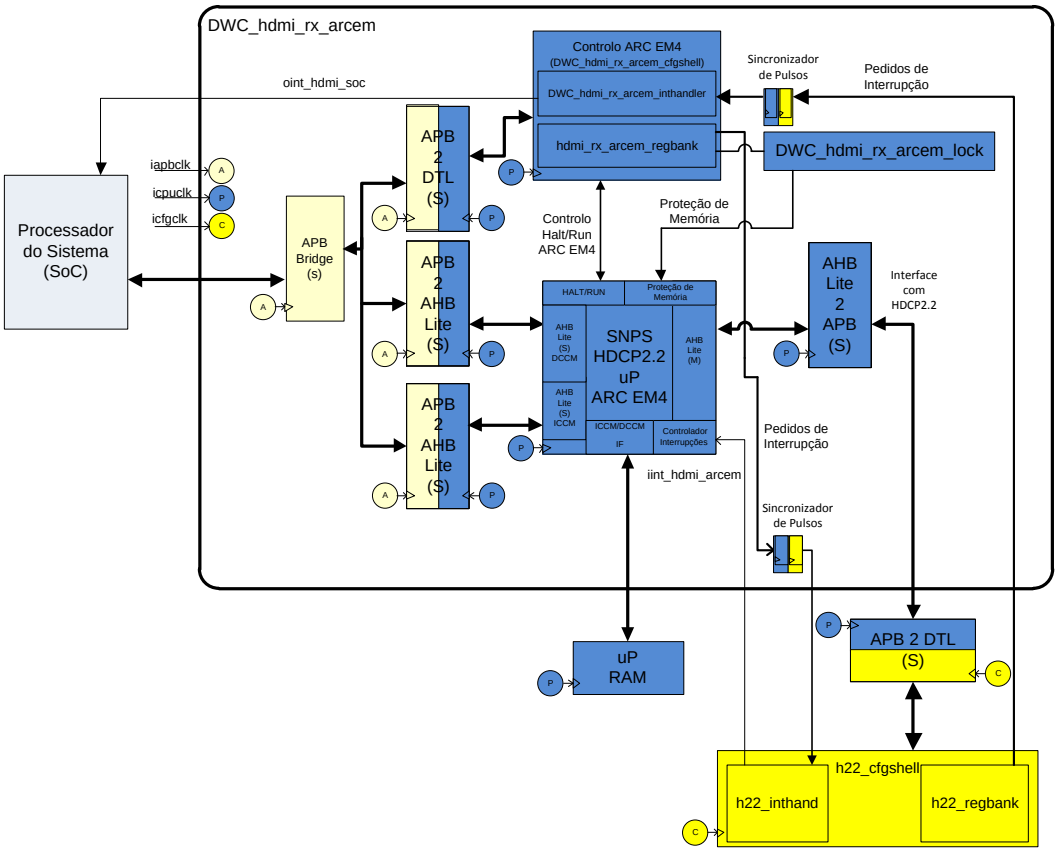

Figura 4.2: Diagrama de blocos do módulo *DWC_hdmi_rx_arcem*

ARChitect2 desenvolvida pela Synopsys. Neste *Software* é possível acrescentar e retirar funcionalidades ao processador assim como definir quais as interfaces que vão estar disponíveis. Aqui definem-se também o tamanho das memórias, número de interrupções entre outros aspetos importantes. No final da configuração, a ferramenta gera um conjunto de ficheiros Verilog que é então incluído no módulo *DWC_hdmi_rx_arcem*.

Para esta aplicação são incluídas duas memórias, uma de instruções (ICCM - *Instruction Closed Coupled Memory*) e outra de dados (DCCM - *Data Closed Coupled Memory*). É também incluído nesta configuração a componente de interrupções que vai permitir ao sistema um melhor controlo do acesso as memórias como explicado anteriormente. A memória ICCM é o local onde estarão todas as instruções relativas ao *Software* que permitirá o controlo do fluxo de informação entre o processador do sistema e o protocolo SNPS HDCP2.2. Já a memória DCCM para além de incluir os dados que o *Software* necessita, é o local onde vão estar contidas as *MailBoxes*, referidas anteriormente, que servirão para a troca de informação entre o processador do sistema e o processador ARC EM4. Estas memórias irão ser implementadas através de *Software*, ou seja, o compilador de *Software* necessita de reservar o espaço necessário para as *MailBoxes* assim como não permitir a alocação de memória na região reservada para as mesmas. Cada uma das *MailBoxes*

Módulo	Descrição
SNPS HDCP 2.2 uP ARC EM4	Este módulo é implementado usando um processador ARC EM4, o qual vai ser usado para controlar todo o fluxo de dados relativos ao SNPS HDCP2.2.
Controlo ARC EM4	Este módulo é responsável por três funcionalidades distintas. Controlar todo o fluxo de interrupções, controlar e monitorizar o estado do processador ARC EM4 e por fim gerir o mecanismo de proteção de memória presente no processador.
APB Bridge	Este módulo tem a finalidade de reencaminhar todas as operações de escrita e leitura para diversos blocos dependendo do endereço que está presente na entrada.
APB 2 DTL Bridge	Este módulo consiste na conversão de uma interface (AMBA APB3) em outra interface (DTL), assim como na conversão de domínios dos relógios.
APB 2 AHB-Lite Bridge	Este módulo consiste na conversão de uma interface (AMBA APB3) em outra interface (AMBA AHB-Lite), assim como na conversão de domínios dos relógios..
AHB-Lite 2 APB Bridge	Este módulo consiste na conversão de uma interface (AMBA AHB-Lite) em outra interface (AMBA APB3).

Tabela 4.1: Lista dos bloco que compõem o módulo *DWC_hdmi_rx_arcem*

será constituída por três campos. Um dos campos indicará o tipo de comando a ser transmitido, haverá outro para indicar o tamanho que os dados associados ao comando vão ocupar e por fim um o terceiro campo onde vão ser colocados os respetivos dados.

Relativamente às interfaces de comunicação com as memórias ICCM e DCCM assim como o acesso aos periféricos, o processador ARC EM4 utiliza os protocolos AMBA AHB-Lite ou BVCI. Para esta aplicação foi escolhido o protocolo AMBA AHB-Lite uma vez que pertence ao grupo AMBA, o mesmo que o protocolo APB3, mantendo assim uma uniformidade nos protocolos. De referir que estas interfaces com as memórias são *slaves*, já o caso da interface com os periféricos é do tipo *master*, ou seja é o processador que controla a comunicação. Neste sistema os periféricos correspondem aos registos do bloco SNPS HDCP2.2, uma vez que irá ser o processador ARC EM4 que irá controlar todo o protocolo SNPS HDCP2.2.

O processador ARC EM4 permite que o acesso as memórias possa ser efetuado a qualquer altura uma vez que o mesmo tem uma prioridade maior que o resto das operações internas ao processador. Apesar desta propriedade, decidiu-se usar a arquitetura com interrupções, como já foi referido, por uma questão de cortesia. Sem este fluxo de pedidos e repostas para aceder às *MailBoxes*, poderia acontecer que o processador ARC EM4 fosse interrompido durante uma operação crítica, o que comprometeria a sua atividade.

O processador ARC EM4 que será utilizado neste sistema está configurado com a opção de *"Halt on Reset"* que faz com que o processador não comece a operar após um *reset* ao sistema. Com esta opção, o utilizador pode integrar *Software* no processador através das respetivas inter-

faces, para que o mesmo arranque da forma pretendida. O controlo de *Halt/Run*, isto é, colocar o processador em modo de espera ou em atividade, é feito através da receção de dois sinais, que correspondem a pedidos de execução ou de espera. Quando o processador recebe algum destes sinais, responde com uma confirmação e entra no estado correspondente. Estes pedidos são realizados externamente pelo bloco de controlo que irá ser detalhado na Secção 4.4.

Existe também uma interface composta por 16 bits que define que regiões de memória estão protegidas contra qualquer tipo de operação. Esta proteção também é conduzida pelo bloco de controlo do processador e será explicada na secção correspondente.

4.4 Controlo ARC EM4

Este módulo, designado por *DWC_hdmi_rx_arcem_cfgshell*, é responsável por grande parte do controlo de todo o sistema. Contém uma interface DTL (*Device Transaction Level*) para que possa ser acedido pelo utilizador e as suas saídas são maioritariamente sinais de controlo e interrupções.

Dentro deste bloco estão inseridos três subsistemas compostos por alguns registos e alguma lógica associada, sendo que cada um destes conjuntos desempenha uma função específica.

4.4.1 Controlo do Processador ARC EM4 e geração de pedidos de interrupções

Neste conjunto estão definidos os registos que controlam o estado do processador, isto é, colocar o processador em modo de espera ou em execução. É também neste grupo que estão os registos responsáveis por gerarem pedidos de interrupções, que vão ser reencaminhados para outro bloco, onde irá ser ativada uma linha de interrupção dependo do pedido efetuado.

4.4.2 Gestor das interrupções provenientes do processador ARC EM4

Neste grupo estão definidos registos que controlam as interrupções provenientes do processador ARC EM4 que têm como destino o processador do sistema. Este subsistema contém um vetor de interrupções, o qual é controlado por três aspetos: por um pedido de interrupção proveniente do processador ARC EM4 e por duas configurações (*Enable* e *Status*).

Os registos aqui presentes definem o valor do *Enable* e *Status* para que o utilizador possa controlar livremente as interrupções. Para que a linha de interrupção seja ativada é necessário que o *Enable* e o *Status* estejam ativados, sendo que o *Status* pode ser ativado pelo utilizador ou por um pedido de interrupções. O esquema representado na Figura 4.3 esclarece a arquitetura aqui usada.

Os vetores de *Status* e *Enable* são vetores que são controláveis através de registos específicos.

4.4.3 Controlo do mecanismo de proteção de memória do processador ARC EM4

O processador ARC EM4 contém um mecanismo de proteção das memórias a ele associadas. Todo o espaço de memória é dividido em 16 regiões, cada uma delas controlada por um bit. Caso esse bit esteja ativado, o processador não permite que se efetue qualquer tipo de leitura ou escrita

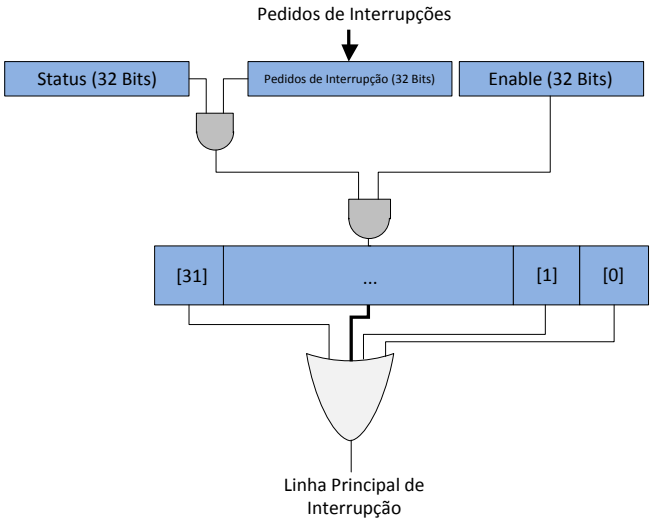

Figura 4.3: Esquema do modo de operação das interrupções

na região correspondente. Neste caso os dados retornados são sempre 0x00 e é acionada uma exceção em *Software*. Para um melhor controlo destes bits de proteção, existe um registo que quando ativado bloqueia qualquer alteração nos bits de proteção, sendo necessário efetuar um *reset* ao sistema para se poder alterar novamente. Na secção 4.11 está descrita uma lista completa de todos os registos existentes, assim como a sua função.

4.4.4 Diagrama de Blocos

Este módulo é composto por cinco sub-blocos com funcionalidades distintas. Toda a estrutura do mesmo está representada na Figura 4.4.

Um dos sub-blocos, nomeadamente o *DWC_hdmi_rx_arcem_fsm*, implementa uma máquina de estados que controla todo o resto do módulo *DWC_hdmi_rx_arcem_cfgshell*. Na figura 4.5 é possível ver o diagrama da máquina de estados que implementa as operações de escrita e leitura assim como condições de erro. Os sinais exibidos nas linhas entre os vários estados representam a condição necessária para a máquina de estado avançar para o respetivo estado. Todos estes sinais pertencem à interface DTL. Já os sinais representados por baixo de cada estado são os sinais que são ativados aquando a entrada no respetivo estado. Os sinais principiados por *odtl* pertencem à interface DTL, ao passo que os sinais começados por *wdtl* informam o resto dos sub-blocos se estamos perante uma operação de escrita ou leitura.

Uma vez que os registos correspondentes ao controlo do processador ARC EM4 e os registos relativos às interrupções estão em blocos separados, é necessário colocar um descodificador de endereços. Este descodificador está representado pelo sub-bloco *DWC_hdmi_rx_arcem_addrdec* e controla, com base no endereço de entrada, qual dos sub-blocos irá receber a operação de escrita/leitura.

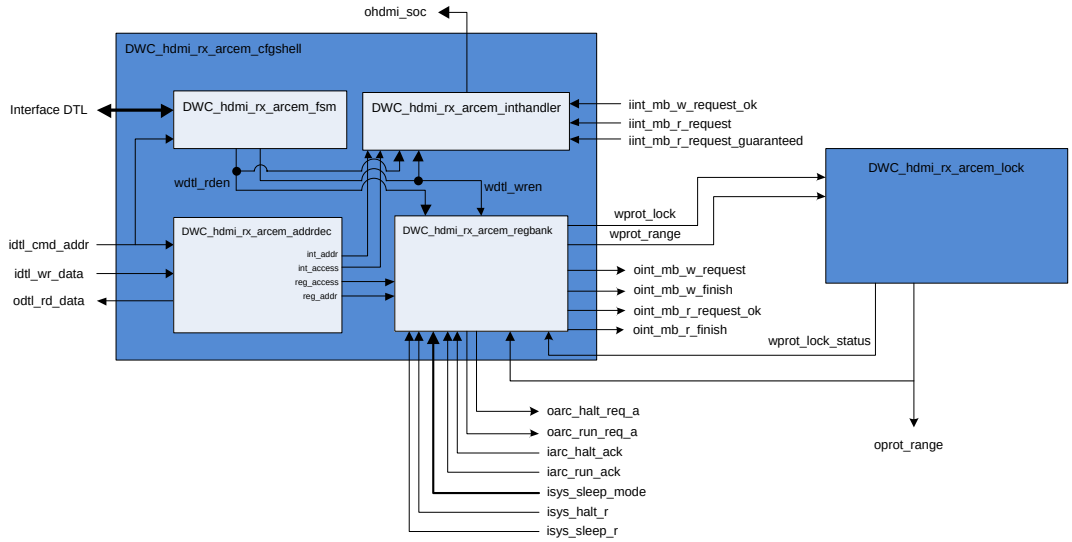

Figura 4.4: Diagrama de blocos do módulo *DWC_hdmi_rx_arcem_cfgshell*

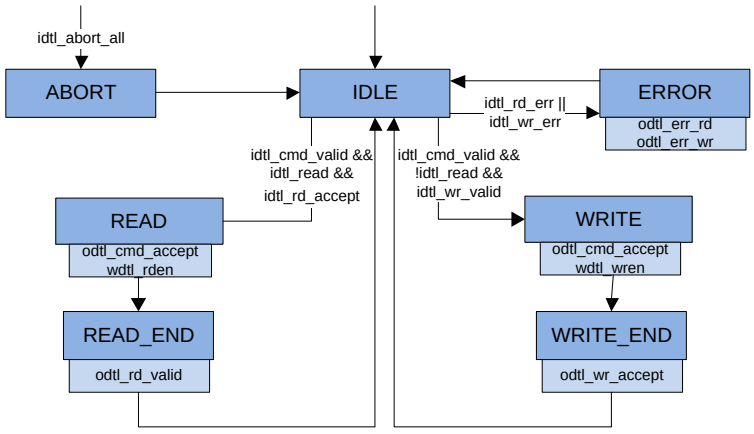

Figura 4.5: Diagrama de blocos do módulo *DWC_hdmi_rx_arcem_cfgshell*

Como referido anteriormente, existem neste bloco registos que controlam as interrupções cujo destino é o processador de sistema e outros registos que controlam o processador ARC EM4. Estes registos estão representados pelos sub-blocos *DWC_hdmi_rx_arcem_inthandler* e *DWC_hdmi_rx_arcem_regbank* respetivamente. Dentro de cada sub-bloco está embutida alguma lógica que permite a realização de operações de escrita e leitura.

O controlo relativo à proteção das memórias é efetuado pelo bloco *DWC_hdmi_rx_arcem_lock*. Este bloco coloca na saída *oprot_range* o valor que o utilizador pretender, mas depois de ser efetuado um bloqueio na proteção, esta saída mantêm um valor constante independentemente do valor que utilizador defina. Apesar de anteriormente se ter referenciado que este sistema de proteção

era separado dos outros sistemas, efetivamente os registos que controlam este mecanismo estão no bloco *DWC_hdmi_rx_arcem_regbank*. Isto acontece porque esses mesmos registos seguem uma estrutura idêntica ao resto dos registos, ficando assim uma disposição de registos mais organizada. Estes registos acedem ao bloco *DWC_hdmi_rx_arcem_lock* através das ligações *oprot_lock* e *oprot_range* representadas na Figura 4.4.

4.5 APB Bridge

Este módulo tem a finalidade de reencaminhar as operações AMBA APB3 para diversos *slaves* com base no endereço que está presente á entrada do bloco. O princípio de funcionamento deste bloco é baseado num multiplexador, o qual está representado na Figura 4.6. Todos os sinais de entrada são replicados para a saída selecionada, o mesmo acontece com os sinais de reposta do protocolo AMBA APB3.

Este bloco, apesar de reencaminhar as operações para uma determinada saída, é capaz de interromper a operação caso o endereço presente na entrada não pertença às gamas de endereço previamente definidas.

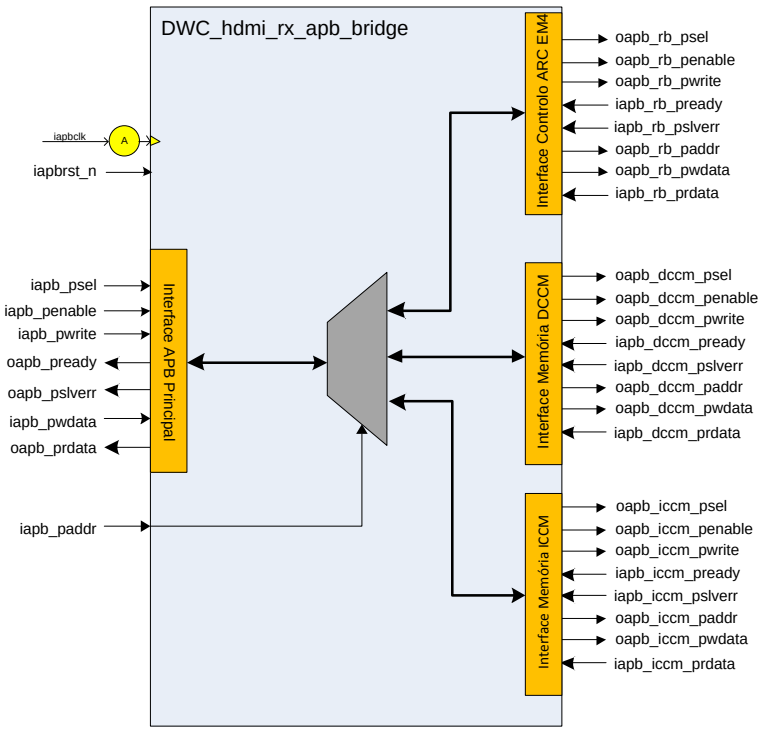

Figura 4.6: Estrutura do módulo *DWC_hdmi_rx_arcem_apb_bridge*

Este módulo irá servir para reencaminhar as operações de leitura e escrita, provenientes do processador de sistema, pelos vários locais de memória/registos, isto é, as memórias do processador ARC EM4 e os registos do bloco de controlo apresentado anteriormente.

4.6 APB 2 DTL Bridge

O bloco de controlo do processador ARC EM4 utiliza uma interface DTL para manter uma uniformidade com os blocos de registos da atual solução do HDMI RX (Recetor HDMI). Dado que a interface no módulo principal é do tipo AMBA APB3, é imprescindível existir um conversor de interfaces para que o utilizador possa comunicar com o bloco de controlo. O mesmo acontece para os domínios de relógio visto que a interface AMBA APB3 utiliza um relógio diferente do que é utilizado no bloco de controlo.

Uma vez que o módulo do HDMI RX (Recetor HDMI) já utiliza um conversor deste género, o mesmo foi acrescentado a este projeto para manter uma coerência. Este conversor utiliza um bloco de controlo com recurso a uma máquina de estados, representada na Figura 4.7, para gerir os sinais de controlo das duas interfaces. O endereço e os dados de escrita e leitura são diretamente conectados de uma interface até à outra.

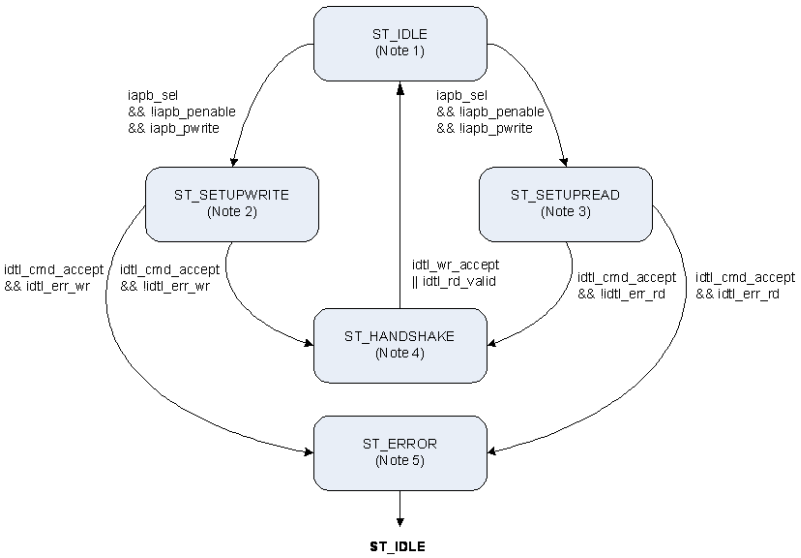

Figura 4.7: Máquina de estados do bloco *DWC_hdmi_rx_arcem_apb2dtl*

Os sinais exibidos nas linhas entre os vários estados representam a condição necessária para a máquina de estado avançar para o respetivo estado. Estes sinais pertencem às entradas da interface AMBA APB3 e DTL. Inicialmente este bloco está num estado de repouso, sendo que com a ativação de uma operação, o mesmo segue para um estado que corresponde ao tipo de operação (escrita ou leitura). Com base nos sinais de resposta da interface DTL, a máquina de estados prossegue para um estado de erro ou de conclusão da respetiva operação, sendo que no fim volta para o estado de repouso.

4.7 APB 2 AHB-Lite Bridge

Uma vez que a interface às memórias do processador ARC EM4 são realizadas através do protocolo AMBA AHB-Lite, é necessário existir um conversor entre a interface AMBA APB3 e

AMBA AHB-Lite. O mesmo acontece para os domínios de relógio, o processador é controlado por um relógio, já a interface AMBA APB3 é controlada por outro domínio de relógio. Este módulo realiza essas duas conversões para que possa ser possível a comunicação com as memórias do processador.

Este bloco contém uma máquina de estados (representada na Figura 4.8) que opera no domínio do relógio do processador ARC EM4 e que controla toda a operação do protocolo AMBA AHB-Lite. Os dados de escrita, de leitura, o indicador de operação de escrita e o indicador de erro são guardados num *buffer* específico para esta transação.

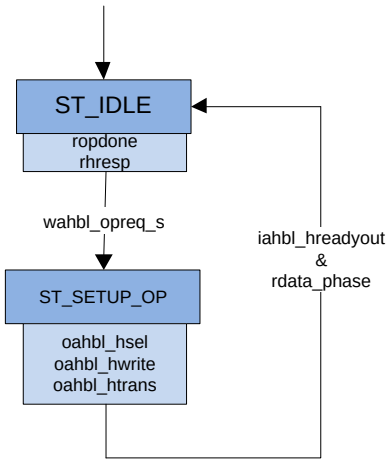

Figura 4.8: Máquina de estados que controla o módulo AMBA APB3 2ahblite

No domínio do relógio que controla a interface AMBA APB3 está incluída uma lógica que realiza o protocolo AMBA APB3. Quando este controlo determina que uma operação foi corretamente invocada, lança um pulso para o bloco que contém a máquina de estados, sendo que passará primeiro por um sincronizador de pulsos. Este pulso irá ser o sinal de controlo que iniciará a máquina de estados.. Uma vez que o protocolo AMBA APB3 é mais lento que o AMBA AHB-Lite e não suporta envios em *burst* (envios consecutivos), os sinais HBURST e HSIZE ficam fixos aos valores binários 000 e 010 respetivamente. Toda a estrutura deste bloco está apresentada na Figura 4.9 onde é possível observar as interligações entre os vários subsistemas que compõem este módulo.

4.8 AHB-Lite 2 APB Bridge

Este módulo tem a finalidade de converter a interface para AMBA APB3 e vai ser usada para fazer a conexão entre o processador ARC EM4 e o banco de registos do SNPS HDCP2.2. Este bloco já existe no portfólio da Synopsys e permite transmissões em modo *burst* ou seja, várias operações consecutivas utilizando para isso várias memórias FIFO, uma vez que o protocolo AMBA APB3 é mais lento que o AMBA AHB-Lite.

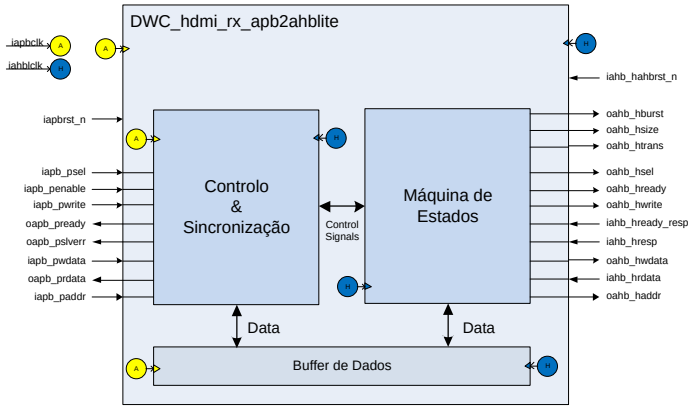

Figura 4.9: Estrutura do módulo *DWC_hdmi_rx_arcem_apb2ahblite*

4.9 Arquiteturas alternativas

Aquando a projeção deste módulo foram equacionadas várias soluções para o mesmo problema principalmente no modo de como as interrupções iam ser processadas.

A primeira hipótese passava por colocar todo processamento de interrupções no controlador de interrupções do bloco SNPS HDCP2.2, sendo deste bloco que sairiam as duas linhas de interrupções (para o processador ARC EM4 e para o processador do sistema). Isto diminuía a área ocupada pelo módulo *DWC_hdmi_rx_arcem*, mas implicava um número muito maior de operações. Quando uma interrupção chegava ao processador do sistema, este tinha de usar as *MailBoxes* para saber que tipo de interrupção é que lhe foi enviada, implicando o uso de mais interrupções ainda, o que poderia levar até um ciclo infinito. Por esta razão, esta hipótese foi descartada.

Uma vez que o problema da solução anterior era o facto do processador do sistema não conseguir ler diretamente qual a interrupção que foi gerada, foi pensada em encaminhar as interrupções individuais até ao bloco de controlo e só nesse momento gerar a linha de interrupção principal. Deste modo o utilizador já conseguia saber que interrupção lhe foi enviada. Apesar desta melhoria ainda surgia uma questão que despendia algum tempo no fluxo de operações, uma vez que a interrupção tinha de ser desativada. Com esta implementação, só o processador ARC EM4 podia ativar e desativar as interrupções cujo destino era o utilizador. Assim sendo, sempre que o processador do sistema recebia uma interrupção e por consequente verificava o seu motivo, o mesmo tinha de pedir ao processador ARC EM4 para a desativar, o que continuava a envolver um grande número de operações extra.

Com esta dificuldade, o melhor a fazer seria disponibilizar ao utilizador a capacidade de verificar que interrupção lhe foi enviada assim como desativar a mesma. A solução para isto é a atual arquitetura, que passa por colocar um manipulador de interrupções completo no bloco de controlo do processador ARC EM4, como explicado anteriormente. Com isto, o aumento de área não é tão significativo e leva a um número mais reduzido de operações.

4.10 Espaço de Memória

Como mencionado anteriormente, este módulo irá ser constituído por vários locais onde é guardada informação. Para uma melhor organização de toda a memória disponível no bloco, foi criado um espaço de endereçamento específico para esta implementação.

Existem espaços de memória que são acessíveis apenas por uma das entidades, mantendo assim uma maior segurança. As gamas de endereços a qual pertencem cada uma dos espaços de informação estão representadas na Figura 4.10.

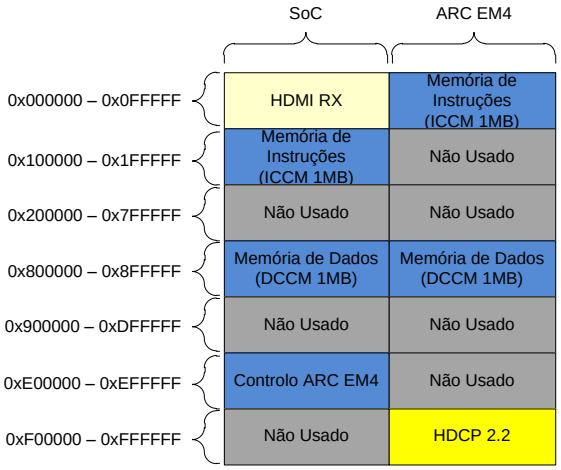

Figura 4.10: Mapeamento da memória utilizado no bloco *DWC_hdmi_rx_arcem*

Apesar dos endereços presentes neste sistema serem de 32 bits, não é necessário utilizar todo o intervalo de memória que os endereços permitem. Posto isto, foram utilizados apenas os 24 bits menos significativos dos endereços. De seguida, o intervalo de endereçamento foi dividido em 16 regiões de 1MB, sendo que algumas dessas regiões têm uma finalidade específica.

Uma vez que os registos relativos ao HDMI RX (Recetor HDMI) estavam previamente definidos para o espaço 0x000000 até 0x0FFFFF, optou-se por manter-los no mesmo intervalo para que não seja necessário qualquer tipo de conversão de endereços. Este espaço de memória é apenas acessível pelo processador do sistema.

Devido a um erro detetado na ferramenta de configuração do processador ARC EM4, não foi possível incluir a memória ICCM na região seguinte. A solução encontrada passa por inserir a memória de instruções na primeira região no espaço de memória visto pelo processador ARC EM4, mas na segunda região no espaço visto pelo utilizador, dado que a primeira região já está ocupada pelos registos do HDMI RX. Assim quando o processador do sistema quer realizar uma operação de escrita ou leitura na memória de instruções, utiliza a gama de endereços apresentada no lado esquerdo da figura 4.10. Internamente, o endereço é convertido para a gama apresentada do lado direito da Figura 4.10. Desde do fim da gama de endereços da memória de instruções, existe um espaço que não é usado, sendo que no futuro este espaço pode ser usado para uma possível expansão da memória de instruções, caso seja necessário.

Para o caso da memória de dados (DCCM), apenas era possível usar a região de memória iniciada pelo endereço 0x800000. Como esta memória é acessível tanto pelo processador de sistema como pelo processador ARC EM4, foi utilizado o endereço base de 0x800000 nos dois casos. O espaço não utilizado a seguir à memória DCCM não é usado mas pode ser definido para uma possível expansão da memória de dados, tal como acontece com memória de instruções.

O bloco representado por *ARC EM4 Control* contém os registos de controlo do processador que apenas são visíveis ao utilizador, uma vez que será ele que indicará quando o processador ARC EM4 estará em execução ou em espera.

Por fim, está representado o espaço de memória dedicado para os registos do bloco SNPS HDCP2.2. Como é possível observar na Figura 4.10, o processador de sistema não tem acesso a estes registos, o que era expectável uma vez que o objetivo deste projeto é retirar o acesso do processador do sistema aos registos do SNPS HDCP2.2.

Sempre que o processador de sistema ou o processador ARC EM4 tenta aceder a um endereço que não está a ser usado, é retornado um erro, cancelando assim a operação.

4.11 Descrição dos Registos

Depois de explicada a distribuição da memória, é tempo de apresentar os registos existentes no sistema e explicar a sua finalidade. Existem dois tipos de registos, aqueles que são apenas de leitura e aqueles que são de escrita. Qualquer tentativa de escrita em registos de leitura ou então a leitura em registos de escrita é retornado um erro. Os registos de escrita podem ainda ter um caso especial, que quando são escritos com o nível lógico 1, os mesmos trocam para o nível lógico 0 após um ciclo de relógio, gerando assim um pulso.

Nas tabelas seguintes são indicados e explicados os registos que foram inseridos em cada bloco.

4.11.1 Controlo ARC EM4

Neste bloco estão inseridos os registos que controlam o estado do processador ARC EM4 assim como os registos que enviam os pedidos de interrupção correspondentes aos acessos à *Mail-Boxes*. Estes registos estão representados na Tabela 4.2.

É também neste bloco que estão os registos que controlam as interrupções que vão para o processador do sistema. Como explicados anteriormente existem dois aspetos que são controláveis e observáveis: *Enable* e *Status* das interrupções. Na Tabela 4.3 estão apresentados os registos que permitem alterar e observar os valores do *Enable* e *Status*.

Por fim, este bloco permite também alterar a proteção das memórias do processador ARC EM4. Os registos que controlam essa mesma proteção são apresentados na tabela 4.4.

O utilizador pode alterar o valor do vetor de proteção assim como bloquear toda a proteção através dos registos *arcem_prot_range* e *arcem_lock* respetivamente. Uma vez que estes registos são apenas de escrita, não é possível obter o seu valor atual. Para que o utilizador possa ler o atual

Endereço	Nome do Registo	Acesso	Descrição
0xE00100	core_info	Leitura	Informação sobre o processador
0xE00104	arcem_sys_status	Leitura	Estado do processador ARC EM4
0xE00108	arcem_req	Escrita	Pedidos de *Halt&Run*
0xE0010C	arcem_ack	Leitura	Respostas a pedidos de *Halt&Run*
0xE00110	arcem_mb_w	Escrita	*MailBoxes*- Pedidos de Interrupção (Escrita)
0xE00114	arcem_mb_r	Escrita	*MailBoxes*- Pedidos de Interrupção (Leitura)

Tabela 4.2: Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell*

Endereço	Nome do Registo	Acesso	Descrição
0xE00200	hdmi_soc_ien_clr	Escrita	Desativar *Enable*
0xE00204	hdmi_soc_ien_set	Escrita	Ativar *Enable*
0xE00208	hdmi_soc_iclr	Escrita	Desativar *Status*
0xE0020C	hdmi_soc_iset	Escrita	Ativar *Status*
0xE00210	hdmi_soc_ists	Leitura	Estado do registo de *Status*
0xE00214	hdmi_soc_ien	Leitura	Estado do registo de *Enable*

Tabela 4.3: Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* referente às interrupções

estado do vetor de proteção e verificar se o mesmo está bloqueado, existem dois registos adicionais de leitura que permitem recolher o valor do vetor assim como do bloqueio.

4.11.2 Memória DCCM (Memória de Dados)

A memória de dados do processador é local onde vão estar alocadas as *MailBoxes* que permitirão a troca de comandos e informações entre o utilizador e o resto do sistema relativo ao protocolo SNPS HDCP2.2.

A maior parte da memória de dados é usada por *Software*, já uma parte da gama de endereços é reservado para as *MailBoxes*, sendo que esta gama irá ser definida pelo utilizador final de acordo com as suas exigências. Neste sistema existem duas *MailBoxes*, uma para pedidos e outra para

Endereço	Nome do Registo	Acesso	Descrição
0xE00300	arcem_lock	Escrita	Bloquear Proteção de Memória
0xE00304	arcem_lock_status	Leitura	Estado do bloqueio da Proteção de Memória
0xE00308	arcem_prot_range	Escrita	Código de Proteção das memórias
0xE0030C	arcem_prot_range_status	Leitura	Código de Proteção das memórias

Tabela 4.4: Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* referentes ao controlo da proteção das memórias

respostas sendo que cada uma delas é composta por três campos. Um campo que indica o tipo de pedido/resposta, outro campo que providencia o tamanho dos dados adicionais que possam vir associados aos pedidos/respostas e por fim um espaço mais extenso onde vão ser guardados os respetivos dados adicionais.

Existe também um espaço dedicado para registos de *debug* que permitirá o utilizador trocar informações de erros assim como outros aspetos importantes.

Estes espaços dedicados para as *MailBoxes* não são definidos em hardware, mas sim pelo utilizador, o que implica que o mesmo tenha de ter atenção no momento da implementação do *Software* a ser inserido no processador. É crucial que o *Software* não utilize este espaço, destinado às *MailBoxes*, para outros afins como por exemplo a declaração de variáveis, uma vez que iria trazer problemas na troca de informações.

4.11.3 Memória ICCM (Memória de Instruções)

A memória ICCM irá ser o local onde estarão inseridas as instruções que constituirão o programa principal que o processador ARC EM4 irá correr. Nesta memória não é necessário reservar à partida qualquer espaço, uma vez que será o compilador de software que fará a gestão do mesmo.

4.11.4 Registos SNPS HDCP2.2

No módulo de controlo do processador ARC EM4, o subsistema *DWC_hdmi_rx_arcem_inthandler* faz a gestão das interrupções que têm como destino o processador do sistema assim como os pedidos de interrupções provenientes do mesmo. Para o sistema ficar completo, é necessário um local que realize a gestão das interrupções cujo destino é o processador ARC EM4, assim como a gestão dos pedidos de interrupções que têm como destino o processador do sistema. Esse local será o banco de registos presente no bloco SNPS HDCP2.2.

Como este módulo já faz parte da atual solução da Synopsys, era muito importante não alterar a sua estrutura e fazer o mínimo de alterações possíveis. A solução encontrada passa por acrescentar os registos correspondentes aos pedidos de interrupções no banco de registos do SNPS HDCP2.2

e adicionar, no vetor de interrupções do mesmo bloco, as interrupções que irão para o processador ARC EM4. Esta implementação é muito idêntica á usada no bloco de controlo do processador ARC EM4. Nas Tabelas 4.5 e 4.6 estão apresentados os registos que foram adicionados ao bloco SNPS HDCP2.2.

Endereço	**Nome do Registo**	**Acesso**	**Descrição**
0xF00300	soc_mb_w	Escrita	*MailBoxes*- Pedidos de Interrupção (Escrita) com destino ao processador de sistema
0xF00304	soc_mb_r	Escrita	*MailBoxes*- Pedidos de Interrupção (Leitura) com destino ao processador de sistema

Tabela 4.5: Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell*

Endereço	**Nome do Registo**	**Acesso**	**Descrição**
0xF00200	hdmi_soc_ien_clr	Escrita	Desativar *Enable*
0xF00204	hdmi_soc_ien_set	Escrita	Ativar *Enable*
0xF00208	hdmi_soc_iclr	Escrita	Desativar *Status*
0xF0020C	hdmi_soc_iset	Escrita	Ativar *Status*
0xF00210	hdmi_soc_ists	Leitura	Estado do registo de *Status*
0xF00214	hdmi_soc_ien	Leitura	Estado do registo de *Enable*

Tabela 4.6: Lista dos registos que estão no bloco *DWC_hdmi_rx_arcem_cfgshell* referente às interrupções

4.12 Fluxo de Operações

A operação básica deste sistema baseia-se no envio e receção de comandos entre o utilizador e o processador ARC EM4. Para a transmissão de comandos são utilizadas *MailBoxes*, isto é, locais de memória pré-alocados, que são acessíveis pelas duas entidades. Para que o utilizador possa enviar um comando, ou seja, escrever na *MailBoxes* de transmissão, é necessário enviar um pedido para o processador ARC EM4 através de uma interrupção. Quando o processador ARC EM4 for notificado da intenção de escrita, o mesmo responde com outra interrupção dizendo que foi facultado o acesso à *MailBoxes*. Para o caso de receção de um comando, o fluxo de operações é semelhante ao apresentado para o envio.

Para qualquer interrupção, é necessário que a entidade recetora desative a mesma, utilizando os registos específicos. Isto é importante porque caso contrário a linha de interrupção mantinha-se ligada para sempre, o que poderia levar a ciclos infinitos em *Software*.

Para uma melhor compreensão do fluxo de operações realizadas no sistema, são apresentados dois fluxogramas onde é exibido a troca de mensagens necessárias para um envio de comando (Figura 4.11) e para a receção de um comando (Figura 4.12).

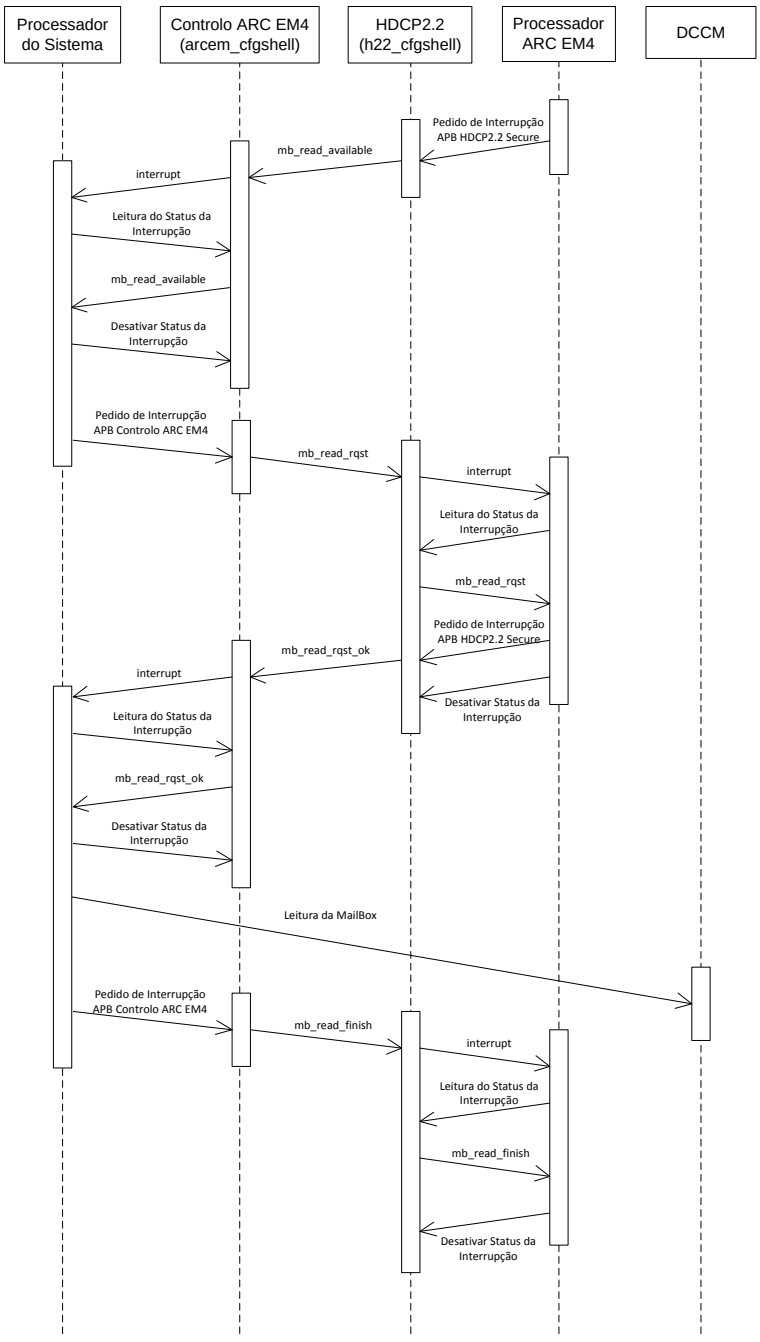

Figura 4.11: Fluxo de operações para o envio de um comando

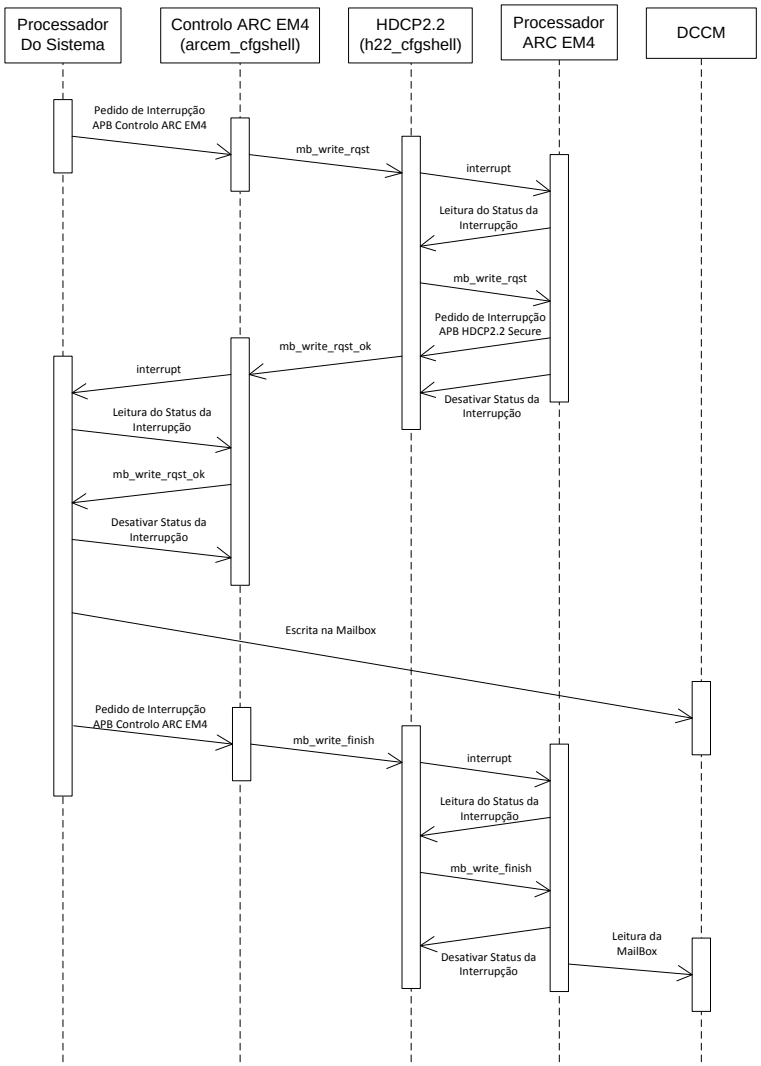

Figura 4.12: Fluxo de operações para a receção de um comando

Com esta arquitetura e com este protocolo de operações, é garantido que não há acessos indesejados às memórias concebendo assim um sistema mais robusto.

De referir que que a linha de interrupção proveniente dos registos do SNPS HDCP2.2 não contem apenas os alertas correspondentes ao acesso às *MailBoxes*. Todas as interrupções que já existiam no sistema SNPS HDCP2.2, continuam a estar presentes. A diferença é que na solução antiga era o processador do sistema que ficava responsável por tratar das mesmas. Com esta nova solução, este esforço é passado ao processador ARC EM4.

4.13 Comandos do sistema

Apesar do objetivo deste projeto não incluir a produção de todo o software para o sistema, é necessário conceber uma estrutura que permita uma fácil integração com qualquer sistema. Por consequente, foi definido um conjunto de comandos que efetua operações básicas no protocolo SNPS HDCP2.2. Estes comandos fazem parte de uma API (*Application Programming Interface*) que permitem ao utilizador final utilizar este sistema de uma forma mais intuitiva e prática. Cada comando é composto por dois campos, o tipo e alguns dados adicionais sendo que este último pode estar vazio caso a operação não necessite de informações extra. Relativamente ao tipo de comando, este é composto por uma palavra de 8 bits, sendo que os 3 bits mais significativos indicam se o comando é um pedido (*Request*) ou uma resposta (*Reply*). Os outros 5 bits, representam o propósito do comando. Nas Tabela 4.7 e 4.8 estão retratados os formatos dos comandos assim como o seu propósito.

Tipo de Comando	**Formato**
Request	3'b101
Reply	3'b010

Tabela 4.7: Bits que indicam o tipo de comando

Com esta definição dos comandos, é possível realizar algumas operações básicas com o protocolo SNPS HDCP2.2. Em cada API existe troca de algumas mensagens específicas entre os dois processadores, assim como operações do processador ARC EM4 no bloco SNPS HDCP2.2. Nos próximas secções estão apresentados a sequencia de passos necessárias para cada operação.

4.13.1 Ligar/Desligar o bloco SNPS HDCP2.2

Se o utilizador deseja ligar ou desligar o motor de encriptação do protocolo SNPS HDCP2.2, então a sequência de passos que a API irá realizar está representada nas Figuras 4.13 e 4.14.

Propósito	Formato	Descrição
ACK	5'b00001	Indica que o pedido anterior foi efetuado com sucesso
HDCP_ON	5'b00010	Ligar o protocolo de encriptação SNPS HDCP2.2
HDCP_OFF	5'b00011	Desligar o protocolo de encriptação SNPS HDCP2.2
HDCP_REAUTH	5'b00100	Reiniciar o protocolo de encriptação SNPS HDCP2.2
HDCP_GET_STATUS	5'b00101	Obter o estado atual do protocolo de encriptação SNPS HDCP2.2

Tabela 4.8: Bits que indicam o tipo de comando

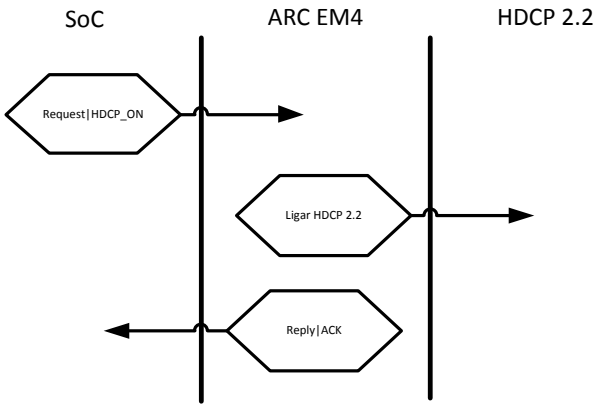

Figura 4.13: Sequência de operações para ligar o bloco SNPS HDCP2.2

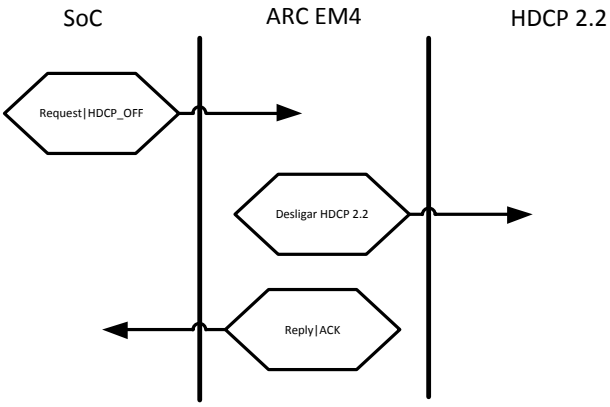

Figura 4.14: Sequência de operações para desligar o bloco SNPS HDCP2.2

4.13.2 Obter o estado SNPS HDCP2.2

No caso de o utilizador desejar obter o estado do motor de encriptação do protocolo SNPS HDCP2.2, isto é, se o mesmo está ligado ou desligar, a mensagem Request|HDCP_GET_STATUS irá ser enviada.

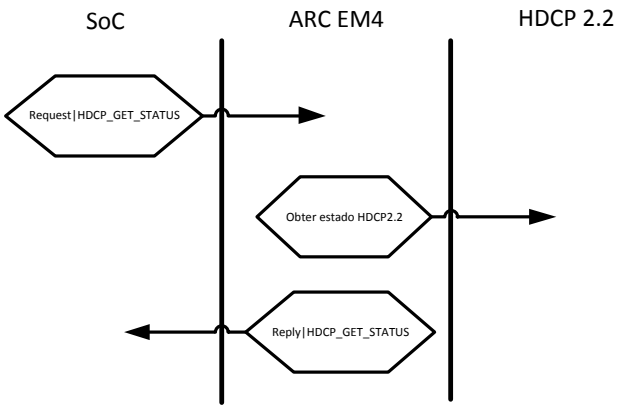

Figura 4.15: Sequência de operações para obter o estado do bloco SNPS HDCP2.2

Neste caso, o processador ARC EM4 terá de realizar uma operação de leitura em registos específicos do SNPS HDCP2.2. Uma vez encontrado o estado atual, o processador responde ao utilizador com a mensagem Reply|HDCP_GET_STATUS juntamente com um dado adicional que corresponde ao atual estado do SNPS HDCP2.2. A sequência de passos que vão ser efetuados está representada na Figura 4.15

4.13.3 Reautenticação do bloco SNPS HDCP2.2

Existe também uma opção que faz com que o bloco SNPS HDCP2.2 reinicie. Esta opção corresponde à reautenticação do motor de encriptação, cuja sequência de passos está apresentada na Figura 4.16.

4.14 Software

Como referenciado anteriormente é necessário incluir no processador ARC EM4 determinado *Software* que permita operar e realizar todas as funções necessárias. Para a construção deste *Software*, irá ser utilizado a linguagem de programação C incluindo algumas bibliotecas do processador ARC EM4 [10]. Para além do código principal é necessário inserir também um *Bootloader*, isto é, um conjunto de instruções que são executas antes do programa principal, que permitem uma configuração inicial do processador.

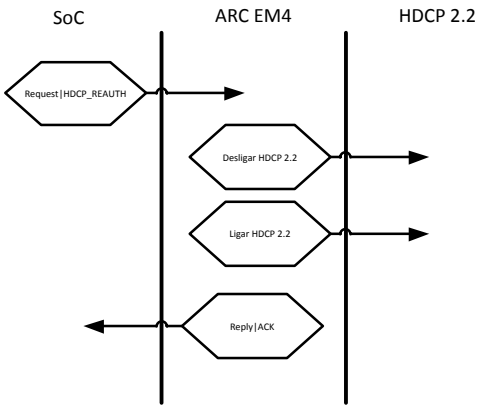

Figura 4.16: Sequência de operações para reautenticar o bloco SNPS HDCP2.2

Todas estas informações são compiladas e enviadas para a memória de instruções através da interface AMBA APB3 que o módulo *DWC_hdmi_rx_arcem* possui. Depois de todo o carregamento da memória, é necessário colocar o processador ARC EM4 em operação. Para isso basta ativar o registo correspondente no bloco de controlo do processador.

Para escrita efetuar uma escrita na interface dos periféricos, isto é, no bloco do protocolo SNPS HDCP2.2 é necessário declarar um apontador com o endereço correspondente (dentro da gama definido anteriormente, isto é, entre 0xF00000 e 0xFFFFFF) e atribuir ao conteúdo desse endereço o valor que se pretende escrever para o bloco SNPS HDCP2.2. Para o caso de uma leitura, efetua-se o oposto, ou seja ler o conteúdo do apontador. Em qualquer um dos casos, o processador ARC EM4 executa o protocolo AMBA AHB-Lite automaticamente, ativando e desativando todos os sinais necessários para que o protocolo seja cumprido.

Para aceder às memórias incluídas com o processador, é possível usar o mesmo princípio usado no acesso aos periféricos, alterando apenas o endereço para o correspondente das memórias. Outro importante aspeto é a implementação de funções que permitam atender interrupções. Isto é conseguido através do uso de funções presentes nas bibliotecas do processador ARC EM4 que associam uma rotina a uma interrupção específica. Um cuidado que é necessário ter na construção do *Software* é o facto de não devem ser instanciadas outras funções dentro da rotina de interrupção.

Estas são as duas funcionalidades que servirão de base para a construção de todo o software necessário para o processador. Associando estas utilidades é possível elaborar a receção e envio de comandos através das *Software* e realizar leituras e escritas no bloco SNPS HDCP2.2 como estava nos requisitos do sistema.

4.15 Resumo

Neste capítulo foi apresentada toda a composição do sistema, expondo e explicando os blocos que constituem o módulo principal, os espaços onde é possível guardar determinada informação,

o fluxo de operações que permitem ao sistema cumprir todos os requisitos impostos e ainda de que forma o software do processador ARC EM4 pode ser construído.

Capítulo 5

Simulação, Verificação e Resultados

5.1 Introdução

Neste capítulo é apresentado todo o fluxo de ferramentas usadas para validar o circuito implementado no capítulo anterior. É explicada de que forma estas ferramentas foram usadas, que configurações foram ajustadas assim como as decisões que foram tomadas. Em cada etapa deste fluxo, são apresentados os resultados que foram obtidos assim como uma explicação dos mesmos.

5.2 Plano de Verificação

Antes da realização de qualquer teste é útil criar um plano de verificação que examine todos os requisitos que o sistema precisa. Este plano contém todos os testes que são necessários realizar para se poder afirmar que o sistema realiza todas as funções que foram inicialmente definidas. Todo este plano pode ser consultado no Anexo A.1, que está presente no fim deste documento.

Os testes estão divididos em 2 grandes grupos: Verificação dos registos de controlo e correta operação do processador. Nos registos de controlo são testadas as operações básicas como a escrita e leituras de registos testando também condições de erro. Posteriormente é analisado as funções de cada bloco específico, isto é, o controlo de interrupções, o controlo da proteção de memórias e o controlo de *Halt&Run* do processador. No que diz respeito aos testes no processador ARC EM4, é avaliado o acesso às memórias por entidades externas, com e sem a proteção de código ativada. Os restantes testes representados no plano de verificação, correspondem a testes que necessitam de *Software* para serem realizados. Uma vez que a concretização do *Software* não está nos objetivos principais deste projeto, nem todos os testes, que pertencem ao processador ARC EM4, foram realizados.

5.3 Simulação e Verificação

Após toda a implementação do módulo apresentado no capítulo anterior, é necessário simular o circuito para verificar se o mesmo cumpre todos os requisitos inicialmente definidos. Para esta

etapa, foram utilizadas ferramentas de simulação da Synopsys, nomeadamente o VCS e o DVE (*Discovery Visual Environment*).

O VCS é uma ferramenta que permite compilar e simular um circuito previamente implementado. Esta ferramenta foi inicialmente usada para verificar se o código Verilog estava correto, isto é, se não havia erros de sintaxe e de estrutura.

Antes de iniciar qualquer verificação foi necessário alterar o código Verilog produzido pela ferramenta de configuração do processador ARC EM4. As memórias DCCM e ICCM estão inseridas internamente no bloco do processador, contudo estes módulos correspondentes às memórias não são sintetizáveis e apenas servem para simulação. Devido a isto, foi alterado o código Verilog de maneira a que o módulo principal possua uma interface dedicada para as memórias, como está representado na Figura 4.1.

Ao iniciar a fase de verificação, foi necessário criar um *testbench* em SystemVerilog, que estimulava o módulo a verificar. Um *testbench* corresponde a um conjunto de instruções em hardware que permite testar um módulo. É neste bloco que são concebidos os sinais de relógio, sinais de *reset* assim como todos os sinais de entrada do circuito. Utilizando também as chamadas ao sistema, é possível imprimir no ambiente de simulação informações acerca do teste que se está a realizar, tais como valor de um determinado sinal, tempos de simulação entre muitos outros aspetos.

Uma vez que o principal ponto de comunicação com o bloco ARC EM4 era a interface AMBA APB3, o *testbench* estimulava sinais que efetuam o protocolo AMBA APB3, tanto para escritas como leituras. O mesmo aconteceu para a interface das interrupções explicada anteriormente.

Toda a verificação do circuito foi fortemente complementada com a visualização das formas de onda produzidas pelo bloco ARC EM4. Isto foi concretizado utilizando a ferramenta de verificação gráfica da Synopsys, designada de DVE (*Discovery Visual Environment*). Os ficheiros resultantes da simulação realizada pelo VCS continham informações acerca das formas de onda de todos os sinais internos e externos ao circuito. Estes ficheiros foram então carregados no DVE, permitindo assim verificar se o circuito funcionava de forma correta.

Durante todo o processo de construção do módulo foram testadas algumas funcionalidades do circuito, começando pelas mais simples e progredindo até as mais complexas. Um dos primeiros testes a serem realizados foi verificar se a interface AMBA APB3 que o circuito dispõe estava a operar de forma correta.

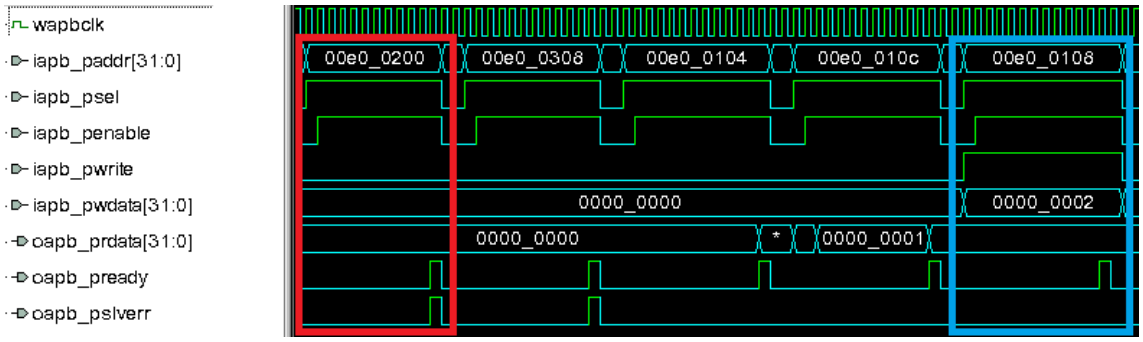

Figura 5.1: Formas de Onda de operações utilizando o acesso AMBA APB3

Na Figura 5.1 estão representados alguns exemplos de operações de escrita e leitura utilizando a interface AMBA APB3. Na região circunscrita por um retângulo vermelho está apresentada uma operação de leitura. Uma vez que esta operação segue o protocolo AMBA APB3, as formas de onda estão de acordo com a Figura 2.2, apresentada na secção 2.4. Esta operação pretende realizar uma leitura no endereço 0xE00200, mas como tal está indicado na Tabela 4.3, este endereço é apenas de escrita, o que leva à ativação do sinal de erro (*PSLVERR*). Na região representada por um retângulo azul, apresenta a situação de uma operação de escrita no endereço 0xE00106.

Outra questão importante a verificar era a entrada e a saída das interrupções. Aqui foram geradas interrupções cujo destino era o processador ARC EM4, o mesmo aconteceu para as interrupções cujo objetivo era notificar o utilizador final. Em ambos os casos foi verificado se as respetivas linhas de interrupções eram ativadas.

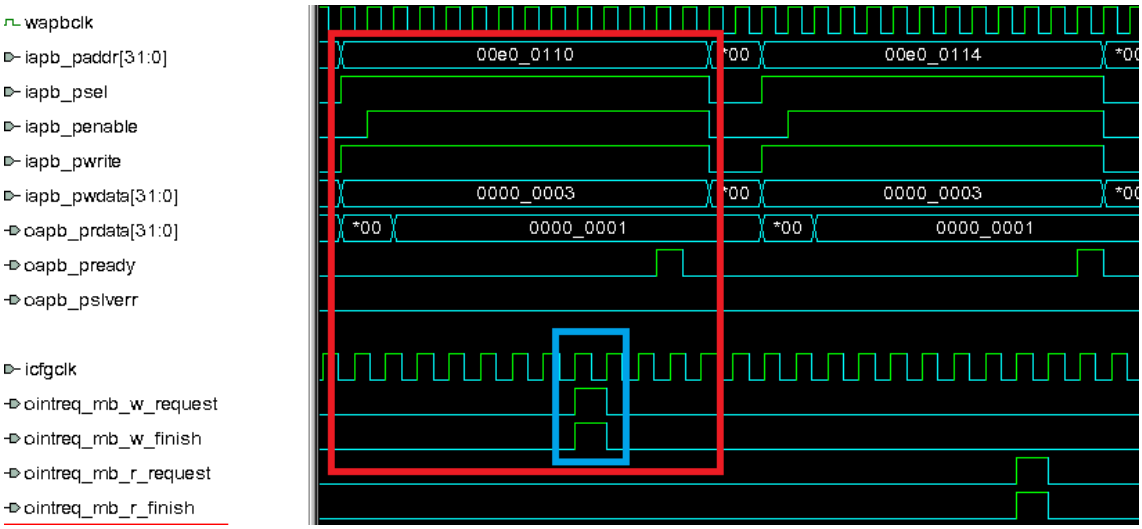

Figura 5.2: Formas de onda de acessos para envios de Interrupções

Na Figura 5.2 estão representadas as formas de onda relativas às ativações das linhas de interrupções. Dentro do retângulo vermelho é possível observar uma operação de escrita nos registos que permitem gerar o pedido de interrupção para o processador ARC EM4. Com a respetiva escrita, é de esperar que os pedidos de interrupção sejam ativados, tal acontecimento está representado pelas formas de ondas que se encontram no retângulo a azul.

Para o caso oposto, onde são os pedidos de interrupção que chegam ao bloco ARC EM4, foi necessário que o *testbench* gerasse esses mesmos pedidos, para além de ativar o *Enable* da linha de interrupção no controlador de interrupções.

Esta simulação está representada na Figura 5.3 e pode ser divida em três partes fundamentais. A primeira corresponde à ativação do pedido de interrupção por parte do *testbench* e consequente ativação da linha principal de interrupções (*oint_hdmi_soc*) que segue para o processador do sistema. Esta fase está representada por um retângulo cinzento na Figura 5.3. Com a receção de uma interrupção, o processador de sistema (neste caso representado pelo *testbench*) necessita de ir verificar qual a razão desta mesma interrupção. Isto é feito através de uma leitura num registo

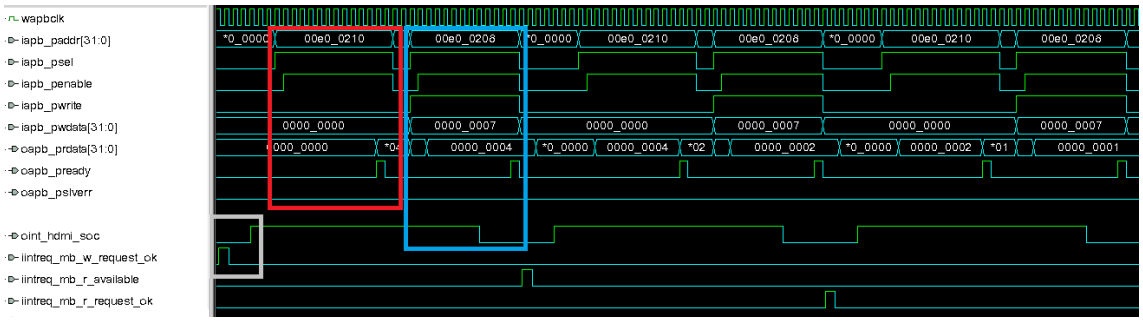

Figura 5.3: Formas de onda na receção de Interrupções

especifico, apresentado na Secção 4.11, e que está representado com um retângulo a vermelho na Figura 5.3. A última etapa, circunscrita por um retângulo azul, representa a desativação da linha principal de interrupção, feita através de uma escrita num registo dedicado para o efeito.

Na Figura 5.3, estão representadas três ativações consecutivas da linha de interrupção principal devido a três pedidos diferentes de interrupções. Para todas elas, o procedimento é sempre igual ao apresentado no parágrafo anterior.

Todos estes testes e estas verificações foram-se acumulando até que se formou o *testbench* principal do circuito. Este *testbench* verifica todos os casos apresentados no plano de verificação, que se encontra no Anexo A.1, de uma forma automática. Quer isto dizer que a própria simulação verifica se os testes foram completados com sucesso sem necessidade de usar um *debugger* como o DVE. Uma vez que este *testbench* é automático, o mesmo gera um relatório contento as operações efetuadas em todos os testes, assim como se algum erro foi detetado.

Outro aspeto muito importante que é necessário verificar é o *Code Coverage*. Este aspeto examina se todas as linhas de código Verilog foram percorridas durante a verificação do módulo. Verifica também se todos os estados das várias máquinas de estados foram percorridos, assim como se foram percorridos todos os casos possíveis das condições.

Aquando a verificação do *Code Coverage*, é expectável que os resultados indiquem que nem todas as condições foram verificadas. Uma razão para isto acontecer é o facto de ser garantido que não existem violações do protocolo AMBA APB3, ou seja qualquer bloco que seja conectado à interface AMBA APB3, irá cumprir todo o protocolo. Mesmo que esta violação aconteça, os blocos mais externos operam de maneira a que essa violação deixe de existir para os blocos mais internos, não permitindo assim um maior *Code Coverage*. Outro aspeto que não apresenta uma percentagem total de *Code Coverage* é a alternância dos valores dos registos. Isto acontece porque existem barramentos de endereços em os bits mais significativos são cortados, uma vez que a entrada do bloco seguinte não necessita de todos os bits.

Como nem todo o *Software* foi implementado, não é possível obter uma grande percentagem de *Code Coverage* no bloco correspondente ao processador ARC EM4 assim como no bloco que converte a interface AMBA AHB-Lite do processador para a interface AMBA APB3 (utilizado para aceder ao bloco SNPS HDCP2.2). Posto isto, não é possível obter uma percentagem geral, ao invés disso foi verificado o *Code Coverage* de cada bloco específico.

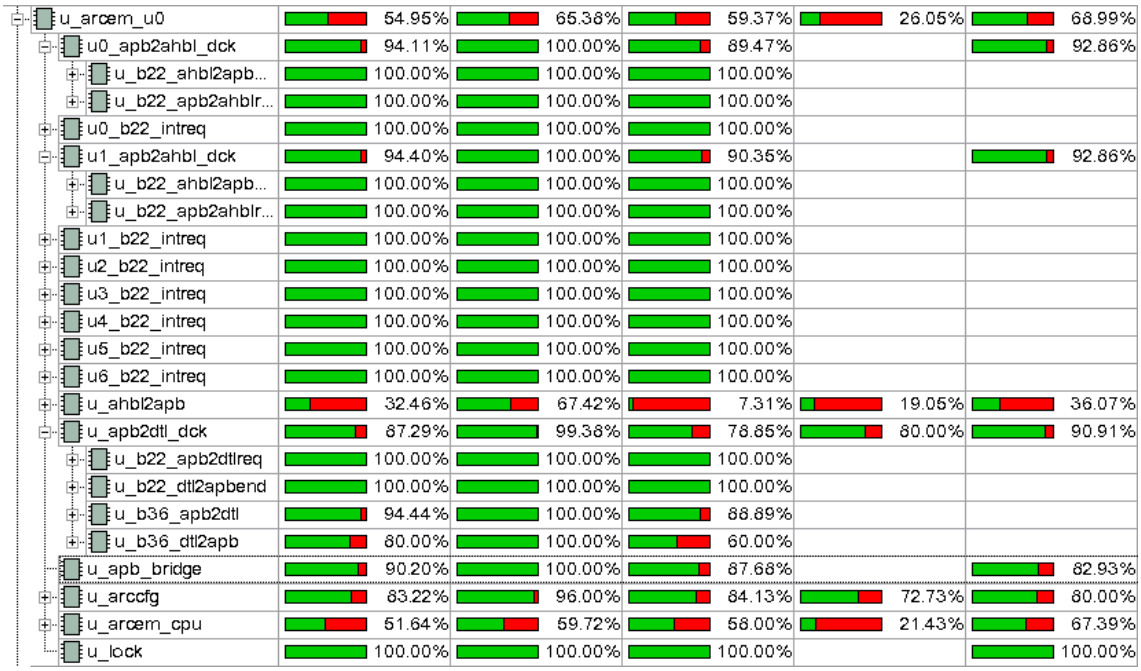

Figura 5.4: *Code Coverage* do módulo *DWC_hdmi_rx_arcem*

5.4 Síntese em ASIC

Após toda a verificação estar concluída, a etapa seguinte é realizar a síntese, isto é a passagem de uma descrição funcional do circuito para uma *netlist* onde estão identificadas todas as ligações elétricas entre as *gates* do circuito. Em primeiro lugar foi realizada a síntese para ASIC utilizando as tecnologias de 40nm e de 28nm. A síntese para FPGA foi apenas tratada posteriormente e será explicada mais a frente neste documento.

Para esta fase foi utilizada a ferramenta *Design Compiler* [11] da Synopsys na sua versão 2013.12-SP5-7. O ambiente de síntese é baseado em comandos, que por uma questão de organização, foram inseridos num script. Este script está dividido em várias partes, cada uma com um objetivo concreto. Inicialmente é configurado os aspetos básicos da síntese como nomes de ficheiros, bibliotecas referentes às tecnologias, ficheiros com configurações relativas ao projeto, entre outros aspetos.

De seguida foram inseridas as restrições temporais referentes ao circuito. Isto implica definir as frequências dos relógios pretendidas, assim como as suas incertezas. Do mesmo modo foram definidos os atrasos máximos e mínimos que possam ocorram nos pinos de entrada e saída do circuito. Os atrasos máximos foram estabelecidos com 40% do período do relógio que controla o respetivo pino, já os atrasos mínimos foram definidos com o valor de 0ns. Estes valores foram escolhidos com base nos valores típicos usados na Synopsys. As incertezas dos sinais de relógio foram definidas com o valor de 0.05ns . O aspeto seguinte, que foi essencial ter em conta, corresponde à inserção de configurações de teste dentro do próprio circuito. Neste caso foi utilizado cadeias de verificação (*Scan Chains*), isto é, uma cadeia que conecta todos os registos do circuito

para uma posterior análise com padrões de teste. Esta funcionalidade é realizada por uma ferramenta dedicada para questões de teste, designada de *DFT Compiler*. Esta ferramenta adiciona algumas funcionalidades ao *Design Compiler* para gerar os registos do circuito com a respetiva cadeia de verificação. O *DFT Compiler* possui o seu próprio método de testes, sendo que faz o mapeamento automático das referidas cadeias de verificação. A Figura 5.5 mostra o fluxo utilizado pelo *DFT Compiler*.

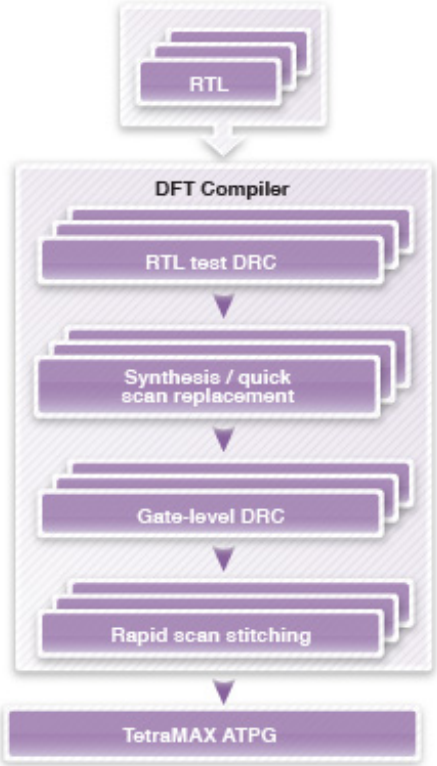

Figura 5.5: Fluxo da ferramenta *DFT Compiler* [4]

Devido à inclusão de cadeias de verificação, foi necessário acrescentar pinos ao módulo principal, pinos estes que correspondem às cadeias de verificação. Foi necessário acrescentar um sinal de relógio dedicado para esta nova característica, uma vez que é este relógio que controlará toda a cadeia de verificação. Também foram adicionados dois pinos (um de entrada e um de saída) para transmissão de dados, para além dos sinais de *Enable* e modo de operação das cadeias de verificação.

Internamente ao módulo principal foram adicionados multiplexadores de relógio com base no sinal que indica o modo de operação das cadeias de verificação. Estes multiplexadores selecionam qual o relógio que vai controlar os registos, se é o relógio originalmente definido ou se é o relógio das cadeias de verificação. Após esta alteração a nível de Verilog, foi importante definir que pinos das cadeias de verificação eram já utilizados internamente no circuito e quais aqueles que a ferramenta de síntese tinha de mapear automaticamente.

A ferramenta *Design Compiler* permite a adição de *Clock Gating* ao circuito. Esta característica possibilita o desligar do sinal de relógio em partes do circuito que não estão a ser usadas. Apesar da existência desta opção, a mesma não foi utilizada para este projeto por diversas razões. Em primeiro lugar porque toda a interface AMBA APB3 assim como os conversores são *slaves*, ou seja, podem receber transferências a qualquer momento, o que implica que estejam sempre ativos. Outra razão é que este módulo é relativamente pequeno em relação ao bloco onde vai ser inserido, o que levou a decisão de não inserir *Clock Gating*. O processador ARC EM4 usado neste projeto já possui duas células de *Clock Gating*, portanto é necessário realizar uma verificação de *Clock Gating*, para confirmar que a ferramenta de síntese assume a existência destas mesmas células como sendo de *Clock Gating*.

O processador ARC EM4 possui uma própria cadeia de verificação, contudo as células (*Clock Gating*) descritas no parágrafo anterior não as detêm. Para tal foi necessário transportar o sinal que define o modo de operação das cadeias de verificação até estas mesmas células e alterar o respetivo código Verilog para que passem a conter a tal cadeia de verificação.

Após todas estas configurações procedeu-se a execução dos scripts, realizando assim todo o processo de síntese. Uma vez concluído, foram verificados os relatórios para conferir se os resultados estavam de acordo com o esperado e se não houve nenhum erro ou violação temporal.

5.4.1 Resultados Temporais

O primeiro relatório extraído do *Design Compiler* verificado corresponde à análise temporal onde estão indicados os caminhos críticos do circuito e o se os mesmo cumprem as restrições impostas anteriormente.

Na Tabela 5.1 estão apresentados os valores de *slack*, isto é, a diferença entre o tempo máximo exigido pelas restrições impostas e o tempo real que os dados demoram a percorrer o caminho. Em caso de resultarem valores positivos, é sinal que os tempos foram cumpridos com a frequência escolhida. Em caso oposto, ou seja, valores de *slack* negativos, indicam que os tempos máximos foram violados e que o circuito não consegue operar à frequência imposta.

	40nm	28nm
FEEDTHROUGH	0.026 ns	115.158 ps
REGIN	1.899 ns	1361.120 ps
REGOUT	0.001 ns	0.364 ps
apbclk	8.356 ns	8538.271 ps
cfgclk	2.772 ns	2909.998 ps
cpuclk	0.914 ns	573.976 ps
scanclk	23.880 ns	24259.395 ps

Tabela 5.1: *Slacks* obtidos na Síntese em ASIC do módulo *DWC_hdmi_rx_arcem*

A primeira coluna indica os caminhos que foram verificados para o calculo dos tempos de propagação. O caso do *feedthrough* representa o caminho crítico entre dois registos do circuito. Relativamente às situações de *regin* e *regout* é representado os caminhos críticos entre uma entrada

e um registo e um registo e uma saída respetivamente. Os restantes casos representam os caminhos que apenas estão num domínio de relógio.

Como é possível observar, todos os valores de *slack* são positivos o que permite afirmar que todos os tempos foram cumpridos. Estes valores resultaram da imposição das frequências apresentadas na Tabela 5.2.

	40nm	28nm
apbclk	100 MHz	100 MHz
cpuclk	150 MHz	250 MHz
cfgclk	300 MHz	300 MHz
scanclk	20 MHz	20 MHz

Tabela 5.2: Frequências de relógio obtidas na Síntese em ASIC do módulo *DWC_hdmi_rx_arcem*

Estes valores foram obtidos através da realização de várias sínteses recursivas, à exceção do relógio *cfgclk* que já estava previamente definido no Guia do utilizador do Recetor HDMI presente na Synopsys.

5.4.2 Área e Consumo de Potência

Posteriormente foram verificados nos relatórios extraídos do *Design Compiler* os resultados de área e de consumo de potência.

	40nm	28nm
Área	45498.50 Portas Lógicas	45053.667 Portas Lógicas
Consumo de Potência	1.2473 mW	0.8849 mW

Tabela 5.3: Área e consumos de potência obtidos na síntese em ASIC

Para calculo da área total do circuito implementado, foi necessário retirar dos relatórios da ferramenta de síntese o valor da área total, assim como o valor da área da porta lógica mais pequena. Os resultados apresentados na Tabela 5.3, significam assim o numero de portas lógicas presentes no circuito.

5.4.3 Cadeias de Verificação

No que diz respeito aos resultados das cadeias de verificação explicadas anteriormente, o cenário é de sucesso, cumprindo todos os aspetos necessários. A ferramenta encontrou uma cadeia de verificação como era espectável, sendo que a mesma aparece como validada, o que indica que a sua inserção foi realizada com sucesso.

5.4.4 *Clock Gating*

Relativamente à verificação de *Clock Gating* foram encontradas duas células, que correspondem às que estão originalmente no processador ARC EM4. Excluindo estas células, não foi encontrada mais nenhuma o que era espectável uma vez que o *Clock Gating* não foi inserido. Isto é confirmado pelo relatório que está representado na Figura 5.6.

```
Clock Gating Summary
------------------------------------------------------------
|    Number of Clock gating elements     |         0         |
|                                        |                   |
|    Number of Gated registers           |     0 (0.00%)     |
|                                        |                   |
|    Number of Ungated registers         |  2814 (100.00%)   |
|                                        |                   |
|    Total number of registers           |     2814          |
------------------------------------------------------------
```

Figura 5.6: Relatório do número de elementos que utilizam *Clock Gating*

Neste relatório é apresentado o numero de registos que afetados por um possível *Clock Gating*, que neste projeto é igual a zero. De salientar que neste relatório não é contabilizado as células de *Clock Gating* inseridas aquando a implementação, como é o caso do processador ARC EM4.

Após a obtenção destes resultados, é adequado dizer que a etapa da síntese está concluída e que se pode passar a próxima fase.

5.5 Validação Formal

Após a síntese estar concluída sem erros nem violações, é tempo de passar para a etapa da validação formal. Esta fase consiste na comparação entre a estrutura do módulo inicial e o resultado da síntese, que apenas resulta em sucesso caso as duas implementações forem iguais e exercerem as mesmas funcionalidades. Para esta etapa é usada a ferramenta *Formality* [12] da Synopsys.

Para esta validação foi utilizado um script base que já usado nos projetos da Synopsys. A este script foi necessário acrescentar duas restrições tanto no circuito pré-síntese como no circuito pós-síntese. Estas restrições dizem respeito aos pinos de *Enable* e do modo das cadeias de verificação, que foram colocados constantemente no valor lógico 0. Isto permite que a ferramenta não encontre problemas na inserção das cadeias de verificação, uma vez que elas não estavam no circuito original.

O resultado do relatório foi de sucesso, o que indica que os dois circuitos estão de acordo, podendo assim avançar para a próxima etapa.

5.6 Produção e Análise de Padrões de Teste

A etapa seguinte corresponde à produção de padrões de testes, para as cadeias de verificação anteriormente inseridas, e posterior análise desses mesmos padrões. Para esta fase é usada a ferramenta da Synopsys, *TetraMax*, que gera padrões de teste para análise de falhas. Esta ferramenta irá analisar todas as falhas, juntamente com aquelas que são inacessíveis ou não testáveis. Na Figura 5.7 está representado o fluxo de passos efetuado pelo *TetraMax* [5].

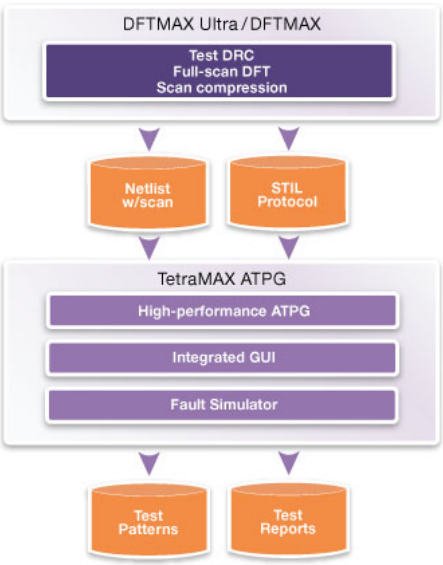

Figura 5.7: Fluxo da ferramenta *TetraMax* [5]

Os resultados obtidos nesta etapa representam a percentagem de falhas possíveis de serem localizadas relativamente ao número de total de falhas que o circuito poderá ter. Como era de esperar, os resultados não são iguais para as duas tecnologias usadas uma vez que os circuitos resultantes das sínteses não são iguais. Os resultados desta ferramenta estão representados nas Tabelas 5.4 e 5.5. Podemos classificar as falhas em quatros tipos diferentes. As que são detetáveis, isto é que são possíveis de serem localizadas. De seguida temos as indetetáveis que representam as que a ferramenta não conseguir encontrar. As não testáveis pela geração automática de padrões são aquelas que a ferramenta encontrou mas apesar disso não as conseguiu testar. Por fim existem as não detetáveis que retratam as falhas que são apenas controláveis ou observáveis e não os dois aspetos em simultâneo.

Na Tabela 5.4 estão representadas as falhas que possam existir no circuito referentes aos casos em que os valores ficam sempre constantes, ou seja, presos. É possível observar que a percentagem de cobertura é bastante elevada, quer isto dizer que quase todas as falhas que possam vir a acontecer no circuito são detetáveis, facto que é desejável. Já no caso do transitório (Tabela 5.5), a percentagem é um pouco mais baixa, mas ainda assim elevada. Este caso representa as falhas que acontecem quando um valor lógico é alterado indevidamente, ou seja, sem qualquer alteração dos valores adjacentes. Este modelo verifica as falhas que ocorrem quando as transições demoram mais tempo do que o previsto devido a erros de fabrico.

Tipo de Falha	Nº de falhas (40nm)	Nº de falhas (28nm)
Detetável	153347	161891
Indetetável	416	417
Não Testáveis por ATPG	40	43
Não Detetável	101	119
Total de Falhas	153904	162470
Cobertura de Teste	99.91%	99.90%

Tabela 5.4: Falhas do tipo *Stuck*

Tipo de Falha	Nº de falhas (40nm)	Nº de falhas (28nm)
Detetável	130114	137991
Indetetável	385	398
Não Testáveis por ATPG	6316	6864
Não Detetável	2549	2533
Total de Falhas	139364	2533
Cobertura de Teste	93.62%	93.62%

Tabela 5.5: Falhas do tipo *Transient*

Com estes resultados podemos concluir que os mesmos estão dentro de uma gama aceitável o que permite dar por concluída esta etapa.

5.7 Análise Temporal

A última etapa na validação do módulo implementada passa por verificar novamente se os tempos são compridos, usando desta vez outra ferramenta da Synopsys designada de *PrimeTime*. Esta ferramenta utiliza um maior numero de fatores e propriedades físicas no cálculo dos tempos de propagação, obtendo assim uma maior precisão dos resultados. Pelos relatórios gerados por esta ferramenta é possível observar que os tempos continuam a ser cumpridos. Tanto numa tecnologia como noutra não foram encontradas violações do tempo de *Setup* dos *flip-flops*, o que era desejado. Relativamente aos tempos de *Hold* dos *flip-flops*, foram reportadas algumas violações. Isto não é preocupante uma vez que estas violações podem ser retiradas aquando a implementação física.

5.8 Síntese para FPGA

Até ao momento foi explicado e apresentado todo o fluxo necessário para a realização da síntese em ASIC. A próxima secção explica todo o processo imprescindível para a concretização da síntese em FGPA, efetuada pela ferramenta *Synplify* da Synopsys. O primeiro passo a realizar-se foi a inserção dos ficheiros Verilog na ferramenta por ordem hierárquica, isto é, do *top-level* até ao bloco de mais baixo nível. Para além disto foi necessário também definir alguns aspetos relativos

à FPGA, à qual se destina este projeto (Virtex7 da Xilinx). Posteriormente foram inseridas as restrições relativas aos atrasos presentes nas entradas e saídas do circuito assim como as frequências pretendidas para os relógios.

Relógio	**Atrasos I/O**
apbclk	1.25 ns
cfgclk	2 ns
cpuclk	5 ns

Tabela 5.6: Restrições temporais das entradas e saídas utilizadas para a síntese em FPGA

Foi utilizado um processo recursivo na determinação da frequência máxima para a qual o circuito cumpria todos os tempos de acordo com as restrições impostas, obtendo as frequências apresentadas na Tabela 5.7.

Relógio	**Frequência**	***Slack***
apbclk	100.0 MHz	8.713 ns
cpuclk	83.3 MHz	6.035 ns
cfgclk	150.0 MHz	1.734 ns

Tabela 5.7: Frequências obtidas na síntese em FPGA

Na Tabela 5.7 estão também representados os valores de *slack* para cada relógio, isto é, a diferença entre o tempo que um sinal tem disponível para atravessar um caminho lógico (imposto pelas restrições e pelas frequências dos relógios) e o tempo que o sinal demora na realidade a percorrer. Como os valores resultantes da síntese são positivos então é possível concluir que todos os tempos foram cumpridos.

Relativamente à área ocupada pelo módulo *DWC_hdmi_rx_arcem* na FPGA, os resultados obtidos estão representados na Tabela 5.8.

	Área
Lookup Tables	7445 (1%)
Número de Registos	3578 ($\approx$ 0%)

Tabela 5.8: Área ocupada pelo módulo *DWC_hdmi_rx_arcem* na síntese em FPGA

Como o módulo *DWC_hdmi_rx_arcem* é relativamente pequeno em relação ao tamanho total da FPGA usada, é de esperar que os resultados de área ocupada sejam pequenos, o que confirmado pela Tabela 5.8.

Para as memórias, a abordagem foi diferente da utilizada na síntese para ASIC. Uma vez que a FPGA possui setores dedicados para efeitos de memória, foi necessário criar duas memórias (DCCM e ICCM) dentro dos padrões da FPGA mas com uma interface que coincidisse com a que processador ARC EM4 oferece.

Depois de concluída a síntese, seguiu-se para a etapa do *Place & Route*, que consiste na alocação de todos os componentes que compõe o módulo dentro da FPGA e posterior ligação

entre os mesmos. Uma vez que apenas nesta fase se conhece os tempos reais de propagação dos sinais, é necessário verificar novamente se os tempos são cumpridos, ou seja, se os valores de *slack* não são negativos.

Numa primeira análise dos relatórios foi verificado que os tempos não eram cumpridos, o que indicou que era necessário realizar alguns ajustes. A solução encontrada passou por criar um módulo de alto nível que engloba todo o módulo ARC EM4, colocando todos as entradas e saídas registadas. De seguida definiu-se, dentro do ficheiro de restrições, que os caminhos até a estes registos não eram analisados pelo *Place & Route*, não contabilizando assim os atrasos que poderiam existir entre as entradas e saídas do módulo e os pinos físicas da FPGA. Esta solução apenas foi possível porque o módulo tem como objetivo ser integrado noutro circuito maior, ou seja não serão as entradas e saídas deste circuito que vão ser ligadas a componentes externos.

Depois de implementada esta solução, realizou-se novamente todo o processo de síntese e de *Place & Route*. Desta vez os tempos foram cumpridos o que indica que a solução foi corretamente implementada. Após esta fase, a síntese para FPGA ficou finalizada ficando assim todo o trabalho de validação terminado. Com a conclusão das sínteses, este projeto ficou concluído, estando validado e sintetizado para as tecnologias enunciadas nos objetivos.

5.9 Resumo

Neste capítulo foi apresentado todo o fluxo usado na validação do módulo implementado. Inicialmente foram apresentados os métodos utilizados na simulação do circuito e os resultados destas mesmas simulações. Após uma verificação de que o circuito operava de forma correta, seguiu-se para a fase de síntese contemplada com todas as etapas posteriores que validam o circuito. Em todas estas etapas foram extraídos e apresentados os resultados com uma explicação dos mesmos.

Capítulo 6

Conclusão e Trabalhos Futuros

6.1 Principais Conclusões do Trabalho

Com o termo do trabalho e com uma análise dos resultados apresentados no capítulo anterior, pode-se concluir que todo o projeto foi realizado com sucesso. Os resultados estão de acordo com as expectativas iniciais, isto é, proceder a integração do novo processador para que este realize todo o processamento relativo ao protocolo SNPS HDCP2.2.

Inicialmente foi feita uma especificação do módulo a implementar de acordo com requisitos iniciais. Devido a algumas adversidades encontradas durante a implementação, foi necessário alterar alguns aspetos que estavam inicialmente pensados. Depois de realizada a construção do modelo Verilog do respetivo módulo, passou-se para a etapa da simulação.

Analisando os resultados das simulações é possível verificar que o módulo implementado efetua todas operações para o qual foi especificado, tendo sido criado um *testbench* que verificava todos os casos de uso definidos num plano de verificação previamente construido.

Após as simulações que confirmavam o correto funcionamento do circuito, passou-se à fase de síntese e validação do circuito. Todos os resultados obtidos estão apresentados e explicados no Capítulo 5. Estes resultados demonstraram que todas as etapas foram concluídas com sucesso, permitindo assim afirmar que todo o projeto foi corretamente validado e está pronto para ser integrado.

Com a conclusão da implementação deste módulo, o perímetro de segurança foi aumentado como era perspetivado no inicio do projeto, assim como foi também retirado carga de processamento ao processador central e por consequente uma maior simplicidade para o utilizador final.

6.2 Sugestões para Trabalhos Futuros

Apesar dos objetivos deste projeto estarem concluídos e cumpridos, existem alguns aspetos que podem servir de base para trabalhos futuros. O primeiro aspeto que surge logo à partida é a implementação do *Software* para o processador ARC EM4. De forma a que o sistema fique

totalmente operacional, é necessário conceber o *Software* que irá controlar todo o acesso ao protocolo SNPS HDCP2.2. Isto inclui também a parte relativa às *MailBoxes* assim como o controlo de interrupções, não só do protocolo SNPS HDCP2.2, mas também do acesso às *MailBoxes*.

É necessário que o compilador do *Software* reserve as regiões de memória da DCCM dedicadas para as *MailBoxes*, para que o utilizador final não utilize essas mesmas regiões para outros fins, podendo assim criar conflitos.

Outra questão importante, e que pode ser vista como um trabalho futuro, é a adaptação deste sistema para outras aplicações. Não só em protocolos que estejam contidos na interface HDMI, mas também noutro tipo de sistemas que necessitem de um aumento do nível de segurança, assim como de uma simplificação para o utilizador final.

Anexo A

Plano de Verificação

Group	Item	Component	Item #	Item check
ARC EM4 Control	APB IF	All Registers	1	Check single write transfer
ARC EM4 Control	APB IF	All Registers	2	Check single read transfer
ARC EM4 Control	APB IF	All Registers	3	Check back to back write transfers
ARC EM4 Control	APB IF	All Registers	4	Check back to back read transfers
ARC EM4 Control	APB IF	All Registers	5	Check write transfer followed of read transfer with PSEL signal asserted
ARC EM4 Control	APB IF	All Registers	6	Check if error signal is asserted if the address is not available
ARC EM4 Control	APB IF	All Registers	7	Check if write on Read Only Register triggers errror signal
ARC EM4 Control	APB IF	All Registers	8	Check if read on Write Only Register triggers errror signal
ARC EM4 Control	APB IF	Register Bank	9	Check if the state of ARC EM4 is keept on the registers
ARC EM4 Control	APB IF	Register Bank	10	Check if a request to Halt/Run is send to ARC EM4
ARC EM4 Control	APB IF	Register Bank	11	Check if a interrupt is generated when wrinting on specific registers
ARC EM4 Control	APB IF	Interrupt Handler	12	Check if the inthandler makes a clear/set of enable/status after a SoC write
ARC EM4 Control	APB IF	Interrupt Handler	13	Check if the inthandler triggers the interrupt signal to the SoC
ARC EM4 Control	APB IF	Interrupt Handler	14	Check if the inthandler is sensitive to the input signals of interrupts
ARC EM4 Control	APB IF	Code Protection	15	Check if the status of the protection range is the same of a previous write.
ARC EM4 Control	APB IF	Code Protection	16	Check if the status of the lock before and after a locking write.
ARC EM4 Control	APB IF	Code Protection	17	Check if the status of the is the protection range is different of a previous write after a lock.
SNPS HDCP 2.2 uP	ARC EM2	uProcessor	18	Check if the DMI IF let write/read on DCCM/ICCM memories
SNPS HDCP 2.2 uP	ARC EM2	uProcessor	19	Check if the Code Protection locks the Write/Read operations on DCCM/ICCM
SNPS HDCP 2.2 uP	ARC EM2	uProcessor	20	Check if the uProcessor enters on halt/run mode
SNPS HDCP 2.2 uP	ARC EM2	uProcessor	21	Check if the uProcessor replies with the ack of halt/run
SNPS HDCP 2.2 uP	ARC EM2	uProcessor	22	Check if the uProcessor send the sleep mode
SNPS HDCP 2.2 uP	ARC EM2	uProcessor	23	Check if the uProcessor uses the AHB-Lite interface to write on HDCP2.2 Registers

Tabela A.1: Plano de Verificação usado no módulo *DWC_hdmi_rx_arcem*

Referências

[1] ARM. *AMBA™ 3 APB Protocol Specification*. ARM, 2004.

[2] ARM. *AMBA® 3 AHB-Lite Protocol Specification*. ARM, 2006.

[3] Synopsys. *DesignWare Cores HDMI Receiver Controller*, 2.11a_ea00 edição, jul 2014.

[4] Synopsys. *DFT Compiler, DFTMAX, and DFTMAX Ultra User Guide*. Synopsys, j-2014.09-sp4 edição, mar 2015.

[5] Synopsys. *TetraMax User Guide*. Synopsys.

[6] www.hdmi.org/manufacturer/hdmi_2_0/. http://www.hdmi.org/manufacturer/hdmi_2_0/. (Visited on 02/18/2015).

[7] Synopsys. *High-Definition Multimedia Interface Specification*, 1.4b edição, oct 2011.

[8] Digital Content Protection LLC. *High-bandwidth Digital Content Protection System*, 1.4 edição, jul 2009.

[9] Synopsys. *DesignWare ARC EM Databook*. Synopsys, 5756-032 edição, aug 2014.

[10] Synopsys. *DesignWare ARCv2 ISA Programmer's Reference Manual*. Synopsys, 5755-035 edição, aug 2014.

[11] Synopsys. *Design Compiler User Guide*. Synopsys, sep 2014.

[12] Synopsys. *Formality User Guide*. Synopsys.